Desarrollo de una herramienta de diseño y modelado de bases de datos NoSQL

Development of a tool for designing and modelling NoSQL databases

TRABAJO DE FIN DE GRADO

Rubén Méndez Alarcón

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Junio 2020

Documento maquetado con TEXIS v.1.0.

Este documento está preparado para ser imprimido a doble cara.

Desarrollo de una herramienta de diseño y modelado de bases de datos NoSQL

Development of a tool for designing and modelling NoSQL databases

Memoria que presenta para optar al título de Graduado en Ingeniería Informática

Rubén Méndez Alarcón

Dirigida por el Doctor

Antonio Sarasa Cabezuelo

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Junio 2020

A mis padres y a mi hermana, Patricia

A Cristina, y a «los de siempre»

Agradecimientos

Antes de comenzar el desarrollo de este trabajo, me gustaría dedicar unas palabras de agradecimiento a las personas que me han acompañado durante la elaboración del mismo.

En primer lugar, a mi tutor, Antonio Sarasa, por ser una fuente de inspiración en todos los momentos que mi mente fue un folio en blanco, así como por su plena dedicación y su incansable labor como docente.

Asimismo, también quisiera agradecer a mis padres haberme brindado la posibilidad de llegar hasta aquí. Puede parecer simple, pero sin ellos nada de esto habría sido posible. También me gustaría reconocer la intachable labor de mi hermana mayor, mi gran ejemplo a seguir y mi mejor consejo.

También a Miha, mi fiel compañero de batalla durante los cuatro últimos años, sin cuyo apoyo me habría rendido más veces de las que me gustaría admitir.

Finalmente, a Cristina, que además de apoyarme incondicionalmente y ayudarme a mantener los pies en el suelo, se ha convertido en una excelente compañera de confinamiento por sorpresa.

A todos vosotros, una vez más, gracias.

Resumen

En los últimos años, la necesidad de procesar ingentes cantidades de datos ha mostrado que las bases de datos relacionales presentaban limitaciones para esta tarea. Esto es debido entre otras razones a las necesidades continuas de mayor capacidad de procesamiento que hacen necesario el uso de soluciones distribuidas.

En este contexto, surgen las bases de datos NoSQL. En general, estas nuevas bases de datos se caracterizan porque no utilizan un esquema de organización de la información fijo y se ejecutan de manera óptima en ambientes distribuidos. En este sentido, son más flexibles para almacenar datos menos estructurados o sin estructura.

Si bien existen algunas herramientas que permiten modelar o diseñar estas bases de datos, son bastante escasas y, normalmente, enfocadas a una plataforma NoSQL concreta.

El objetivo de este Trabajo de Fin de Grado es crear un editor web que permita modelar una base de datos NoSQL orientada hacia agregados de datos. El usuario trabajará sobre un editor usando una notación gráfica independiente de un lenguaje específico. Cuando termine el modelado, podrá exportar su trabajo a la sintaxis concreta de MongoDB.

PALABRAS CLAVE

NoSQL, agregado, base de datos, modelado, diseño, editor gráfico

Abstract

Along the last years, the need of processing massive quantity of data has evidenced that relational databases presented some limitations in this scope. This is due, among other reasons, to the continuous necessities of a bigger capacity of processing which lead to distributed solutions.

In this context we talk about NoSQL databases. In general, this new kind of database is charactized by not using a fixed schema for the organization of data. As well, they have a good behaviour in distributed environments. In this way, they are more flexible to strore less structured data, or even data without a structure.

Even though there are some tools for modelling or designing this kind of databases, they are quite rare and normally focused on a specific NoSQL platform.

The purpose of this Final Project is to create a web application which allows to design an aggregate-based NoSQL database. The user will work on an editor using a graphic notation, which is not attached to a specific language. When the design is complete, the user will be able to export the work to the MongoDB syntax.

KEYWORDS

NoSQL, aggregate, database, modelling, design, graphic editor

Índice

Agradecimientos VII

Resumen IX

Abstract XI

1. Introducción **1**
1.1. Motivación 1
1.2. Objetivos 2
1.3. Estructura de la memoria 3

1. Introduction **5**
1.1. Motivation 5
1.2. Objectives 6
1.3. Structure of the report 7

2. Estado del arte **9**
2.1. Introducción 9
2.2. Variety Schema Analyzer 9
2.3. MongoDB Schema Validation 10
2.4. MongoDB Compass 11

3. Especificación de requisitos **13**
3.1. Introducción 13
3.2. Actores del sistema 13
3.3. Casos de uso 13
3.3.1. Casos de uso comunes a todos los actores 14
3.3.2. Casos de uso del usuario del sistema 16
3.3.3. Casos de uso del administrador del sistema 21

4. Tecnologías empleadas **23**
4.1. Introducción 23

4.2. Tecnologías empleadas en la totalidad de la aplicación 23
4.2.1. HTML5 23
4.3. Tecnologías en el lado del cliente 24
4.3.1. Bootstrap 4 24
4.3.2. CSS3 24
4.3.3. JavaScript 24
4.3.4. JQuery 25
4.3.5. AJAX 25
4.3.6. Cytoscape JS 26
4.4. Tecnologías en el lado del servidor 26
4.4.1. Apache HTTP Server 27
4.4.2. MongoDB Community Server 27
4.4.3. PHP 7 27
4.4.4. PHP Mailer 28
4.5. Otros recursos empleados 28
4.5.1. GitHub 28

5. Modelo de datos **29**
5.1. Introducción 29
5.2. Modelo E-R 29
5.3. Implementación 30
5.4. Estructura de la base de datos de la aplicación 30
5.4.1. Colección usuarios 31
5.4.2. Colección modelos 32
5.4.3. Colección pwdRecovery 32

6. Arquitectura de la aplicación **35**
6.1. Introducción 35
6.2. Tipo de arquitectura 35
6.3. Tareas realizadas en el servidor 35
6.4. Tareas realizadas en el cliente 36

7. Diseño de la aplicación **39**
7.1. Introducción 39
7.2. Página de inicio 40
7.3. Registro de usuarios 41
7.4. Login o acceso a la aplicación 42
7.5. Recuperación de contraseña 42
7.6. Mis modelos 44
7.7. Diseñador 46
7.7.1. Añadir item 48

7.7.2. Eliminar item 49
7.7.3. Editar item 49
7.7.4. Reiniciar lienzo 49
7.7.5. Ajustar 50
7.7.6. Deshacer 50
7.7.7. Rehacer 51
7.7.8. Ayuda 51
7.8. Formulario dinámico para la inserción de datos 51
7.9. Guardar y exportar 53
7.10. Mi perfil 54
7.11. Editar perfil 54
7.12. Panel de administración de usuarios 56

8. Evaluación con usuarios **59**
8.1. Introducción 59
8.2. Metodología 59
8.3. Interpretación de los resultados 60
8.3.1. Datos personales 60
8.3.2. Conocimientos de bases de datos 61
8.3.3. Nivel de satisfacción con la aplicación 63
8.3.4. Sugerencias y comentarios 70

9. Conclusiones y trabajo futuro **75**
9.1. Introducción 75
9.2. Conclusiones del trabajo 75
9.3. Trabajo futuro 76

9. Conclusions and future work **77**
9.1. Introduction 77
9.2. Conclusions of the project 77
9.3. Future work 78

A. Guía de instalación de la aplicación **79**
A.1. Introducción 79
A.2. Instalar y configurar el servidor 79
A.3. Instalar intérprete PHP 80
A.4. Instalar MongoDB Community Server 81
A.5. Copiar el código fuente al directorio del servidor 82
A.6. Importar información en la base de datos 82

B. Guía de uso de la aplicación **85**
B.1. Introducción 85

B.2. Página de inicio 85
B.3. Registro de usuarios e inicio de sesión 85
B.3.1. Registro de usuario 85
B.3.2. Login o inicio de sesión 87
B.3.3. Recuperación de contraseña 87
B.4. Creación y gestión de proyectos 91
B.4.1. Crear proyecto 91
B.4.2. Descartar cambios 91
B.4.3. Guardar cambios 92
B.4.4. Cambiar nombre 92
B.4.5. Eliminar modelo 92
B.5. Creación de modelos 94
B.5.1. Añadir item 94
B.5.2. Eliminar item 94
B.5.3. Editar item 95
B.5.4. Reiniciar lienzo 96
B.5.5. Ajustar 97
B.5.6. Deshacer y rehacer 97
B.5.7. Ayuda 98
B.6. Introducción de datos y exportación 98
B.6.1. Introducir datos 98
B.6.2. Modificar y eliminar datos introducidos 98
B.6.3. Exportar resultados 99
B.7. Cuentas de usuario 100
B.7.1. Mi perfil 100
B.7.2. Editar perfil 100
B.7.3. Administración de usuarios 101

C. Preguntas realizadas en la evaluación con usuarios 105
C.1. Introducción 105
C.2. Preguntas del bloque de información personal 105
C.3. Preguntas del bloque de conocimientos técnicos 105
C.4. Preguntas del bloque de satisfacción con la aplicación 105
C.5. Comentarios 106

Bibliografía 115

Índice de figuras

1.1. Ejemplo de agregado que almacena la información de una persona 2

1.1. Example of an aggregate containing a person's information . . 6

3.1. Diagrama de casos de uso comunes 14
3.2. Diagrama de casos de uso de usuario 16
3.3. Diagrama de casos de uso de administrador 21

5.1. Diagrama entidad-relación 30
5.2. Representación gráfica del modelo de datos 31

7.1. Diagrama de flujo ilustrando las secuencias de uso principales 40
7.2. Vista de la página de inicio 41
7.3. Vista de la página de registro de usuarios 42
7.4. Vista de la página de inicio de sesión 43
7.5. Vista de la página de inicio de sesión tras un intento de login incorrecto 44
7.6. Vista de la página de solicitud de nueva contraseña 45
7.7. Vista del formulario para cambiar la contraseña 46
7.8. Vista del listado de modelos del usuario 47
7.9. Vista del diseñador gráfico 48
7.10. Vista del formulario para insertar datos 52
7.11. Vista de la página de guardar y exportar con el código generado 54
7.12. Vista de datos de un usuario en la sección «Mi perfil» 55
7.13. Vista del formulario de modificación de datos personales . . . 55
7.14. Vista del panel de administración de usuarios 56

8.1. Distribución de la edad de los encuestados 61
8.2. Distribución de los encuestados por género 61
8.3. Distribución de los encuestados por nivel de estudios 62

8.4. Distribución de los encuestados en función de si han trabajado con bases de datos o no 62
8.5. Distribución de los usuarios de bases de datos en función de la plataforma que usan habitualmente 63
8.6. Plataformas NoSQL con las que han trabajado los encuestados 63
8.7. Distribución de los problemas típicos que encuentran los usuarios en las bases de datos NoSQL 64
8.8. Distribución de los usuarios respecto a lo sencillo que les parece el procedimiento de registro 64
8.9. Distribución de los usuarios respecto a lo sencillo que les parece el procedimiento de login 65
8.10. Distribución de los usuarios en función de lo intuitivo que creen que es cambiar la contraseña 65
8.11. Distribución de los usuarios respecto a lo intuitiva que les resulta la página «Mi perfil» 66
8.12. Distribución de los usuarios según las dificultades presentadas el editar su perfil 66
8.13. Distribución de los usuarios respecto a lo fácil que les resulta la gestión de modelos 67
8.14. Distribución de los usuarios respecto a lo fácil que les resulta crear un nuevo modelo 67
8.15. Distribución de los usuarios respecto a lo intuitivo que les resulta el diseñador 68
8.16. Distribución de los usuarios respecto a lo fácil que les resulta usar las herramientas del diseñador 69
8.17. Distribución de los usuarios respecto a la dificultad expresada para insertar datos en un modelo 70
8.18. Distribución de los usuarios respecto a la dificultad expresada para modificar datos insertados en un modelo 71
8.19. Distribución de los usuarios según lo intuitiva que consideran la página «Guardar y exportar» 71
8.20. Satisfacción de los usuarios con el código generado 72
8.21. Satisfacción general de los usuarios con el diseño de la aplicación 72
8.22. Satisfacción general de los usuarios con el funcionamiento de la aplicación 73
8.23. Comentarios y sugerencias de los usuarios encuestados 73

B.1. Ubicación de los controles del carrousel en la página de inicio 86
B.2. Formulario de registro cumplimentado con datos de ejemplo . 86
B.3. Formulario de login cumplimentado con datos de ejemplo . . 87
B.4. Resultado de un intento incorrecto de login 88

B.5. Formulario de solicitud de nueva contraseña cumplimentado con datos de ejemplo 88
B.6. Mensaje de confirmación tras enviarse el email para cambiar la contraseña 89
B.7. Contenido del email recibido para cambiar la contraseña . . . 89
B.8. Formulario de cambio de contraseña cumplimentado con datos de ejemplo 90
B.9. Mensaje de confirmación tras cambiar la contraseña 90
B.10. Pasos a seguir para crear un nuevo proyecto o modelo 91
B.11. Pasos a seguir para descartar los cambios sobre un borrador . 92
B.12. Pasos a seguir para guardar un modelo en edición 93
B.13. Pasos a seguir para cambiar el nombre de un modelo 93
B.14. Pasos a seguir para eliminar un modelo guardado 94
B.15. Pasos a seguir para añadir un item al modelo 95
B.16. Pasos a seguir para eliminar un item del modelo 95
B.17. Pasos a seguir para cambiar el nombre de un item 96
B.18. Pasos a seguir para reiniciar el lienzo 96
B.19. Ejemplo de uso del botón «ajustar» 97
B.20. Ubicación de los botones «deshacer» y «rehacer» 97
B.21. Ubicación del botón «ayuda» 98
B.22. Pasos a seguir para introducir datos en un modelo 99
B.23. Pasos a seguir para editar o eliminar datos de un modelo . . . 99
B.24. Ubicación de los botones «guardar cambios» y «copiar al portapapeles» en la página para exportar resultados 100
B.25. Ejemplo de perfil de usuario mostrando la ubicación de los botones «Mi perfil» y «Editar perfil» 101
B.26. Pasos a seguir para modificar la información personal del usuario 101
B.27. Vista del panel de administración 102
B.28. Pasos a seguir para modificar los datos de un usuario 102
B.29. Pasos a seguir para eliminar un usuario 103

C.1. Pregunta sobre el grupo de edad 106
C.2. Preguntas sobre el género y el nivel de estudios 107
C.3. Preguntas sobre la experiencia con bases de datos 108
C.4. Preguntas sobre la experiencia con bases de datos NoSQL . . 109
C.5. Preguntas sobre la usabilidad I 110
C.6. Preguntas sobre la usabilidad II 111
C.7. Preguntas sobre la usabilidad III 112
C.8. Preguntas sobre la usabilidad IV 113
C.9. Preguntas sobre la usabilidad V 114
C.10. Sección habilitada para sugerencias y comentarios 114

Índice de Tablas

3.1. CU01 - Registro de usuario 15
3.2. CU02 - Login (inicio de sesión) 15
3.3. CU03 - Logout (cierre de sesión) 15
3.4. CU04 - Acceder a un modelo existente 17
3.5. CU05 - Crear nuevo modelo 17
3.6. CU06 - Añadir nuevo agregado 18
3.7. CU07 - Modificar un agregado 18
3.8. CU08 - Eliminar un agregado 19
3.9. CU09 - Insertar datos en un modelo existente 19
3.10. CU10 - Eliminar datos de un modelo existente 20
3.11. CU11 - Eliminar un modelo 20
3.12. CU12 - Exportar modelo 20
3.13. CU13 - Modificar usuario 21
3.14. CU14 - Eliminar usuario 22

Capítulo 1

Introducción

Yo realmente no sabía lo que hacía cuando empecé. Acababa de empezar a escribir canciones.

Neil Young

1.1. Motivación

Las bases de datos NoSQL son un nuevo tipo de bases de datos que han surgido en el contexto del fenómeno Big Data como alternativa al modelo relacional. Su popularización se debe a la capacidad que otorgan para almacenar grandes volúmenes de datos con formatos no uniformes, así como la facilidad que presentan para distribuir los datos mediante técnicas de escalado horizontal.

Como en todas las bases de datos, es importante hablar de los modelos que permiten la organización de los datos. En concreto, a la hora de diseñar el modelo de datos de una aplicación se debe tener en cuenta el tipo de datos a tratar, cómo se quieren almacenar y cómo serán consultados esos datos [1].

En el contexto de las bases de datos NoSQL, uno de los modelos más extendidos es la agregación. Un agregado en un conjunto de información estructurada que se manipula como una unidad. Gráficamente podría representarse como una caja dentro de la cual se puede insertar información u otras cajas a su vez.

Existen varias familias de bases de datos NoSQL que organizan sus datos siguiendo un modelo de agregación [2]:

- Bases de datos clave/valor (Redis)

- Bases de datos orientadas a documento (MongoDB)
- Bases de datos basadas en columnas (Cassandra, HBase)

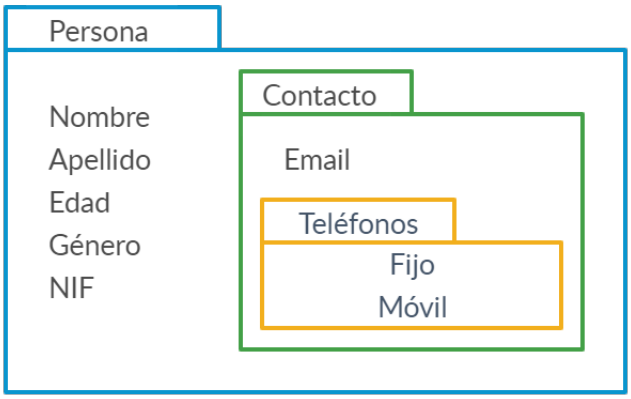

Figura 1.1: Ejemplo de agregado que almacena la información de una persona

Una tarea habitual cuando se trabaja con una base de datos es poder realizar el diseño de la persistencia de la información. Para ello, generalmente se utilizan editores que permiten representar el diseño usando alguna notación gráfica específica como, por ejemplo, la mostrada en la figura 1.1.

El resultado de este trabajo ha sido la creación de una herramienta informática específica para diseñar agregados orientados a bases de datos documentales, en concreto para MongoDB.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es crear una herramienta para modelar bases de datos NoSQL en torno a agregados de una forma sencilla y visual. Este objetivo se particulariza en los siguientes subobjetivos:

- Que se pueda trabajar con la herramienta de una forma genérica, es decir, sin necesidad de conocer la sintaxis de algún lenguaje NoSQL. Para ello, el usuario trabajará con agregados de datos.
- Que una vez generado el modelo, tanto este como las operaciones realizadas sobre el mismo se puedan exportar a una sintaxis particular. En concreto, se exportará como operaciones de MongoDB.

- Que el usuario pueda registrarse y almacenar sus modelos en un servidor web, para verlos de nuevo, editarlos, añadir o eliminar información.

1.3. Estructura de la memoria

Para facilitar la lectura de esta memoria, en esta sección se presenta la estructura en capítulos de la misma, con los contenidos tratados en cada uno.

En el capítulo 1 se plantea una breve introducción al tema, así como las motivaciones para el desarrollo y los objetivos a lograr.

En el capítulo 2 se analiza el estado del arte, que incluye referencias a otras tecnologías ya implementadas con un carácter similar al de este trabajo.

En el capítulo 3 se presenta una especificación software de los requisitos necesarios, elaborada antes de comenzar con la implementación del código.

En el capítulo 4 se introducen todas las tecnologías empleadas en el desarrollo de la aplicación, con el objetivo de familiarizar al lector con las mismas de cara a los capítulos más técnicos.

En el capítulo 5 se desarrolla la estructura del modelo de datos de la aplicación.

En el capítulo 6 se describe con detalle la arquitectura de la propuesta software.

En el capítulo 7 se presenta el diseño de la aplicación programada.

En el capítulo 8 se exponen los resultados obtenidos de una evaluación con usuarios.

En el capítulo 9 se analizan las conclusiones obtenidas del presente Trabajo de Fin de Grado, así como el trabajo futuro que podría desarrollarse sobre el mismo.

Se han incluido tres apéndices:

- En el apéndice A se puede encontrar una guía de instalación de la aplicación a partir del código fuente, para realizar las pruebas necesarias sobre la misma.

- En el apéndice B se adjunta una guía completa de uso de la aplicación.

- Para concluir, en el apéndice C se proporcionan las preguntas realizadas en la encuesta de la evaluación con usuarios.

.

Chapter 1

Introduction

I didn't really know what I was doing when I started. I just started writing songs.

Neil Young

1.1. Motivation

NoSQL databases are a new type of databases that have appeared in the context of the Big Data phenomenon as an alternative to the relational model. Due to the capacity they offer to store extensive volumes of data, as well as the easiness they provide to distribute data with horizontal scaling, their popularity has increased substantially over the last years.

As it happens with all databases, it is important to talk about the models around which the data is organized. When it comes to design an application data model, there are several aspects that must be taken into account, such as the nature of the data, how the data will be stored and how it will be retrieved [1].

In this context of NoSQL databases, one of the most popular strategies for modelling is aggregation. An aggregate is a group of structured information which can be manipulated as unity. Graphically, this could be understood as a box, which can contain information or other boxes.

There are some families of NoSQL databases that organize their data following an aggregation-based model [2]:

- Key-value databases (Redis)
- Document-oriented databases (MongoDB)

- Column-oriented databases (Cassandra, HBase)

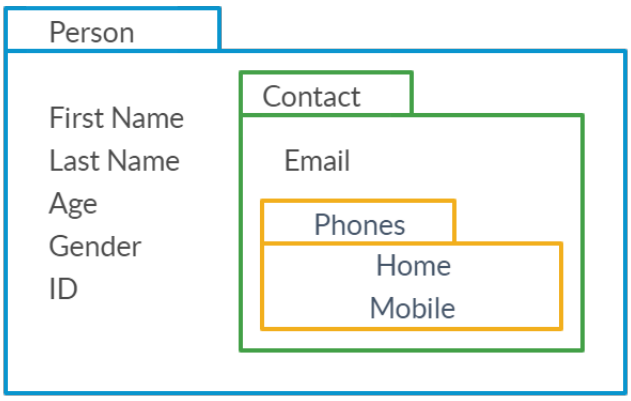

Figure 1.1: Example of an aggregate containing a person's information

A common task when working with a database is creating the design of information persistance. In order to achieve this, it is usual to use editors that allow representing the design as a graphic notation, as it is shown in figure 1.1.

The result of this project is the creation of a computer application. With this tool, the user can design aggregates oriented to document-based databases, in particular, for MongoDB.

1.2. Objectives

The main purpose of this Final Project is developing a tool to model NoSQL databases, in an intuitive and visual way with data aggregates. This objective breaks down into the following sub-objectives:

- The tool can be used in a generic way, in essence, the user does not have to get skills with any NoSQL language. To get this, the user will work with data aggregates.
- Once the model is generated, the model itself and all the operations carried out on it will be exportable. In particular, this will get exported as MongoDB operations.
- The user will be able to sign up and store their models in a web server, in order to open them again, and making changes such as editing the model or putting new data into it.

1.3. Structure of the report

In order to make easier the reading of this report, the structure in chapters will be explained in this section, as well as the contents dealt in each chapter.

Chapter 1 presents a brief introduction to the topic, the motivations for this development and the objectives that should be achieved.

Chapter 2 describes and references some existing tools with similar purposes to the ones of this Project.

Chapter 3 includes a formal software specification with all the requirements of the application.

Chapter 4 deals about the technologies used in the development of the software, in order to get the reader familiar to them, as it will be necessary for understanding the most technical chapters.

Chapter 5 expounds on the application data model.

Chapter 6 presents the architecture of the application.

Chapter 7 collects all the information about the design of the coded application.

Chapter 8 shows the results obtained from an evaluation with users. These results are analyzed in order to perform a better user experience.

Chapter 9 sums up the conclusions made of this Final Project, so as the future work that might be done on it.

At the end of the document, there are three appendixes:

- Appendix A can be useful for installing the application from the source code.
- Appendix B explains in detail how to use the application.
- Appendix C shows the questions that the users were asked during the evaluation.

Capítulo 2

Estado del arte

Todas las verdades son fáciles de entender, una vez descubiertas. El caso es descubrirlas.

Galileo Galilei

En el presente capítulo se pretende hacer referencia a tecnologías similares al objeto de desarrollo de este Trabajo de Fin de Grado. Para cada uno de los ejemplos citados se enumerarán las características principales.

2.1. Introducción

Como se ha comentado en el capítulo anterior, dentro de las bases de datos NoSQL hay distintas familias. Dado que este trabajo se centra en las que se orientan a agregados de datos, todas las herramientas estudiadas están en dicha línea. A continuación se detallan estas tecnologías de interés.

2.2. Variety Schema Analyzer

Variety es una herramienta escrita en JavaScript para MongoDB. Se trata de una solución gratuita, de código abierto, disponible en GitHub[1].

Su función principal es, dada una colección, analizarla en busca de un esquema de organización de los datos. Como resultado, muestra todas las ocurrencias de todas las claves distintas de los documentos [3]. En base a esto, expone información resumida sobre el tipo de datos que ha encontrado asociado a la clave, el número de veces que se ha encontrado y el porcentaje de documentos en el que aparece.

[1] https://github.com/variety/variety

Además, incluye funcionalidades avanzadas que permiten configurar diferentes formas de análisis, tales como buscar solo entre documentos insertados recientementes, o mostrar el último valor asociado a una cierta clave.

Como aspecto negativo de esta herramienta, podría mencionarse que solo es compatible con MongoDB.

2.3. MongoDB Schema Validation

Schema Validation[2] es una de las múltiples opciones que ofrece MongoDB, uno de los sistemas más populares de bases de datos documentales. Se incorpora en la plataforma desde la versión 3.2.

Esta herramienta permite crear esquemas de validación en formato JSON, de tal forma que el administrador de la base de datos pueda decidir un formato normalizado para todos los documentos que se insertan en una determinada colección.

Los esquemas de validación pueden crearse al mismo tiempo que una nueva colección, o bien aplicarse sobre una colección existente a posteriori. Además, Schema Validator permite especificar el grado de tolerancia a los incumplimientos del esquema [4]:

- **Comportamiento estricto**: si un documento incumple el esquema de la colección, se lanza un error y el documento es rechazado.
- **Comportamiento moderado**: si el documento incumple el esquema, se muestra una advertencia para informar de que ese documento no estará «normalizado», pero se permite insertarlo.

Al aplicar un esquema sobre una colección existente con documentos ya insertados, también se tiene en cuenta el grado de tolerancia. Si es moderado, el esquema solo se aplicará a posteriori, es decir, exclusivamente a documentos insertados desde ese momento. Si está configurado en nivel estricto, todos los documentos insertados hasta el momento deberán cumplir el esquema.

No obstante, el administrador de la base de datos puede autorizar a otros usuarios para hacer los llamados *Bypass Document Validation*. En resumen, estos usuarios tendrán permiso en ciertas colecciones para insertar documentos que no cumplan el esquema, incluso si Schema Validator está configurado en modo estricto.

[2] https://docs.mongodb.com/manual/core/schema-validation/

2.4. MongoDB Compass

Compass[3] es una aplicación de escritorio que hace las veces de interfaz gráfica de MongoDB. Permite realizar de una manera visual las operaciones que el usuario ejecutaría en una terminal de MongoDB, como crear colecciones e índices, insertar y editar documentos, así como realizar consultas a la base de datos [5].

Entre sus opciones también se encuentra la validación de documentos, funcionalidad que cubre con Schema Validation (que se explica en el apartado anterior), con un funcionamiento idéntico.

Otras características reseñables de Compass son que admite la inserción de datos en formato CSV y JSON, y que incluye un apartado de estadísticas sobre la base de datos. Su principal inconveniente es que requiere estar dado de alta en Atlas, la plataforma cloud de MongoDB.

[3] https://www.mongodb.com/products/compass

Capítulo 3

Especificación de requisitos

Lo esencial es invisible a los ojos

Antoine de Saint-Exupéry

3.1. Introducción

En este capítulo se identifican los actores que tendrá el sistema. De igual manera, se establecen las interacciones entre el sistema y los distintos tipos de usuarios, recogidas en forma de casos de uso que se detallan a continuación.

3.2. Actores del sistema

En el sistema se han identificado los siguientes actores:

- **El usuario del sistema:** es aquel que hace uso de los distintos componentes del sistema para modelar una base de datos no relacional en base a agregados.

- **El administrador del sistema**: se encarga de realizar todas las gestiones correspondientes a la administración de usuarios: modificar datos de los usuarios, reasignar permisos y eliminar cuentas.

3.3. Casos de uso

A continuación se enumeran y detallan los casos de uso resultantes de la identificación de requisitos, agrupados según los actores que los llevan a cabo.

3.3.1. Casos de uso comunes a todos los actores

En la figura 2.1 se muestran los casos de uso que son comunes a todos los actores de la aplicación. Como puede observarse, se tratan del registro de usuarios, el inicio y el cierre de sesión.

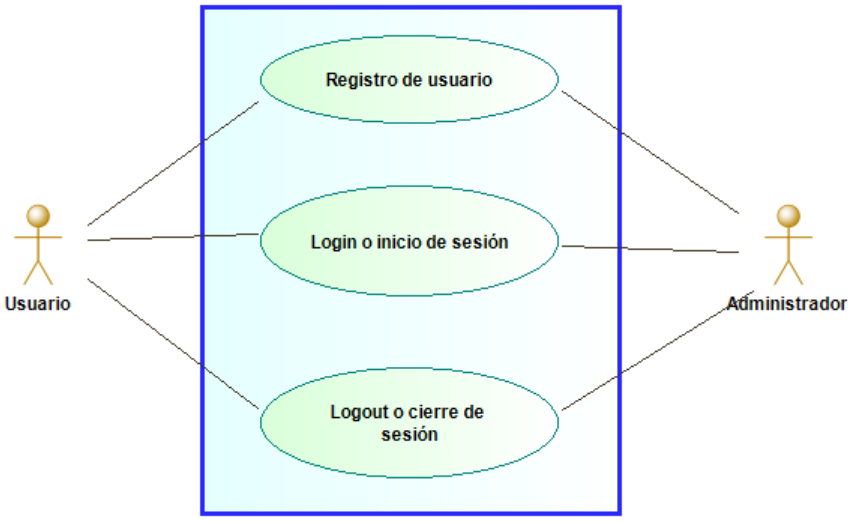

Figura 3.1: Diagrama de casos de uso comunes

En las tablas que siguen se desarrollan los casos de uso identificados.

CU - 01	**Registro de usuario**
Objetivos asociados	Registrar a un usuario en el sistema
Entradas	▪ Nombre ▪ Apellidos ▪ Email ▪ Contraseña
Salidas	Usuario creado
Precondición	▪ La dirección de correo debe ser válida ▪ La contraseña debe tener 5 o más caracteres alfa-numéricos ▪ La repetición de la contraseña debe coincidir
Secuencia normal	1. El usuario accede a la página de registro 2. Introduce sus datos en el formulario de registro 3. Envía el formulario 4. Se inicia su sesión
Postcondición	La sesión debe quedar iniciada

Excepciones	E1. El email indicado ya está registrado E2. La contraseña no es válida o no coincide con su repetición
Comentarios	

Tabla 3.1: CU01 - Registro de usuario

CU - 02	**Login (inicio de sesión)**
Objetivos asociados	Iniciar la sesión de un usuario
Entradas	▪ Email ▪ Contraseña
Salidas	Listado de modelos del usuario
Precondición	▪ El email debe estar registrado ▪ La contraseña debe ser correcta
Secuencia normal	1. El usuario accede a la página de login 2. Introduce sus datos en el formulario de login 3. Envía el formulario 4. Se redirige al listado de modelos del usuario
Postcondición	La sesión debe quedar iniciada
Excepciones	E1. El formato del email introducido es incorrecto E2. El email no está registrado E3. La contraseña no es correcta
Comentarios	

Tabla 3.2: CU02 - Login (inicio de sesión)

CU - 03	**Logout (cierre de sesión)**
Objetivos asociados	Cerrar la sesión del usuario identificado
Entradas	No aplica
Salidas	No aplica
Precondición	El usuario debe estar logueado
Secuencia normal	1. El usuario hace click en «cerrar sesión» 2. Se borran los datos de sesión del usuario 3. Se redirige al usuario a la página principal
Postcondición	Todos los datos de sesión del usuario son eliminados
Excepciones	E1. El usuario no está logueado
Comentarios	

Tabla 3.3: CU03 - Logout (cierre de sesión)

3.3.2. Casos de uso del usuario del sistema

En este apartado se listan y describen los casos de uso que ejecutará el usuario del sistema. Si bien un usuario administrador también podría llevar a cabo estas tareas, no es el objetivo principal del diseño. Como puede observarse en la figura 2.2, estos casos de uso tienen que ver con todo lo relacionado con el modelado de una base de datos NoSQL.

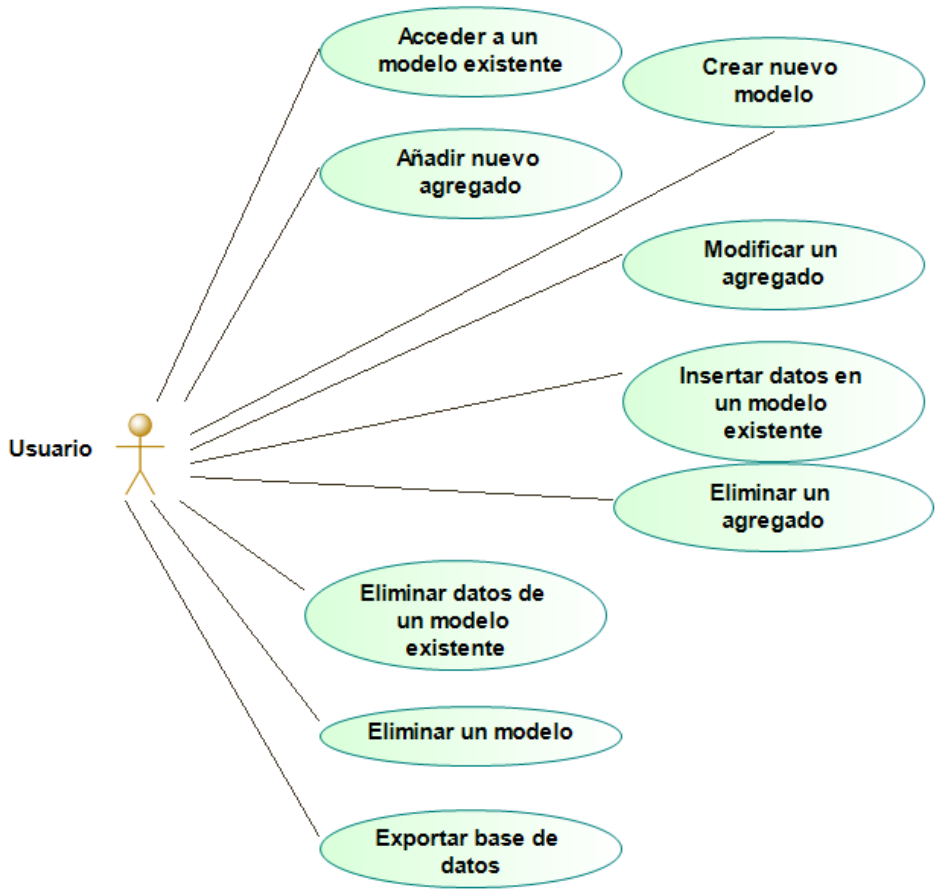

Figura 3.2: Diagrama de casos de uso de usuario

CU - 04	Acceder a un modelo existente
Objetivos asociados	Visualizar o editar un modelo creado anteriormente
Entradas	No aplica
Salidas	Vista del modelo seleccionado
Precondición	▪ El usuario debe estar logueado ▪ El usuario debe tener modelos ya creados
Secuencia normal	1. El usuario hace click en uno de sus modelos 2. Si el usuario ya está editando otro modelo, debe confirmar la acción 3. El usuario puede continuar su trabajo donde lo dejó
Postcondición	El modelo queda cargado y es editable
Excepciones	E1. El usuario no está identificado E2. El usuario no ha creado aún ningún modelo E3. El usuario ha cancelado la acción
Comentarios	

Tabla 3.4: CU04 - Acceder a un modelo existente

CU - 05	Crear nuevo modelo
Objetivos asociados	Diseñar un nuevo modelo
Entradas	Nombre del modelo
Salidas	Modelo vacío (sin ningún agregado)
Precondición	▪ El usuario debe estar identificado ▪ No se debe estar editando ya otro modelo
Secuencia normal	1. El usuario hace click en "Nuevo modelo" 2. Se introduce el nombre deseado en la entrada de texto 3. El usuario hace click en aceptar
Postcondición	Se crea un nuevo modelo vacío
Excepciones	E1. El nombre introducido no es válido (p.e. contiene espacios)
Comentarios	

Tabla 3.5: CU05 - Crear nuevo modelo

CU - 06	Añadir nuevo agregado
Objetivos asociados	Añadir un agregado al modelo actual

Entradas	▪ Selección de ratón del agregado contenedor ▪ Nombre del agregado
Salidas	El agregado se añade a la vista
Precondición	El agregado debe estar contenido dentro de otro agregado (el modelo en sí mismo también se considera agregado)
Secuencia normal	1. El usuario selecciona un agregado existente y hace click en el botón "añadir item" 2. El usuario introduce el nombre deseado para el agregado en la entrada de texto 3. El usuario hace click en aceptar
Postcondición	El agregado creado aparece en el editor gráfico
Excepciones	E1. El nombre introducido para el agregado es inválido (p.e. contiene espacios)
Comentarios	

Tabla 3.6: CU06 - Añadir nuevo agregado

CU - 07	**Modificar un agregado**
Objetivos asociados	Actualizar el nombre de un agregado
Entradas	Selección de ratón con el agregado a modificar
Salidas	Actualización del agregado en la vista
Precondición	Debe haberse seleccionado un agregado
Secuencia normal	1. El usuario selecciona el agregado clickeándolo 2. Hace click en el botón "editar item" 3. Escribe el nuevo nombre del agregado 4. Confirma la operación.
Postcondición	Se actualiza la vista con los cambios realizados
Excepciones	E1. No se ha seleccionado ningún agregado E2. El nuevo nombre introducido no es válido (p.e. cadena vacía)
Comentarios	

Tabla 3.7: CU07 - Modificar un agregado

CU - 08	**Eliminar un agregado**
Objetivos asociados	Eliminar un agregado del modelo
Entradas	Selección de ratón con el agregado a eliminar
Salidas	Actualización de la vista

Precondición	Se debe haber seleccionado un agregado
Secuencia normal	1. El usuario selecciona el agregado que desea eliminar 2. El usuario debe confirmar la operación, sabiendo que se borrará la información asociada a ese agregado en todos los documentos insertados
Postcondición	El agregado y todos sus datos asociados son eliminados del sistema
Excepciones	E1. No se ha seleccionado ningún agregado
Comentarios	

Tabla 3.8: CU08 - Eliminar un agregado

CU - 09	**Insertar datos en un modelo existente**
Objetivos asociados	Asociar documentos a un modelo existente
Entradas	Datos que se van a insertar
Salidas	Actualización de la vista con el documento insertado
Precondición	Debe haber al menos un agregado en el modelo que contega otro agregado
Secuencia normal	1. Se genera un formulario de manera dinámica, atendiendo a la estructura del modelo diseñado 2. El usuario rellena el formulario con los datos que desea insertar 3. Se confirma la operación pulsando el botón «insertar»
Postcondición	La información insertada se refleja dentro del agregado en el editor
Excepciones	E1. No se cumple la precondición E2. El formulario enviado está vacío
Comentarios	

Tabla 3.9: CU09 - Insertar datos en un modelo existente

CU - 10	**Eliminar datos de un modelo existente**
Objetivos asociados	Eliminar un documento asociado a un modelo
Entradas	Documento a eliminar (mediante un click de ratón)
Salidas	Actualización de la vista, eliminando el documento borrado
Precondición	El modelo debe tener al menos 1 documento asociado

Secuencia normal	1. El usuario selecciona el documento a eliminar haciendo click 2. El usuario hace click en «eliminar» 3. El usuario debe confirmar la operación
Postcondición	El documento seleccionado se elimina del modelo
Excepciones	No aplican
Comentarios	

Tabla 3.10: CU10 - Eliminar datos de un modelo existente

CU - 11	**Eliminar un modelo**
Objetivos asociados	Eliminar un modelo y todo su contenido asociado
Entradas	Selección de ratón (click) con el modelo a eliminar
Salidas	Actualización de la vista
Precondición	El modelo debe existir
Secuencia normal	1. El usuario hace click en el botón "eliminar modelo" del modelo a eliminar 2. El usuario debe confirmar la operación
Postcondición	El modelo se elimina del sistema y se actualiza la vista
Excepciones	E1. El modelo no existe
Comentarios	

Tabla 3.11: CU11 - Eliminar un modelo

CU - 12	**Exportar modelo**
Objetivos asociados	Generar código MongoDB para el modelo diseñado
Entradas	Información insertada en el modelo
Salidas	Sintaxis de MongoDB para el modelo indicado
Precondición	El modelo debe existir y contener información
Secuencia normal	1. El usuario selecciona el modelo a exportar 2. El usuario se dirige a la página de exportación 3. El usuario copia el código para utilizarlo en otro sistema
Postcondición	No aplica (el sistema queda en el mismo estado)
Excepciones	E1. El modelo no contiene información
Comentarios	

Tabla 3.12: CU12 - Exportar modelo

3.3.3. Casos de uso del administrador del sistema

En este apartado se describen los casos de uso que corresponden exclusivamente al administrador del sistema.

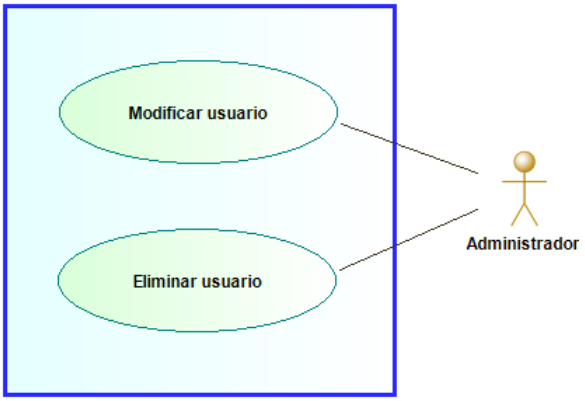

Figura 3.3: Diagrama de casos de uso de administrador

CU - 13	**Modificar usuario**
Objetivos asociados	Modificar la información de registro de un usuario
Entradas	▪ Email actual del usuario a modificar ▪ Datos a actualizar
Salidas	Mensaje de confirmación
Precondición	▪ Necesario estar identificado como **administrador** ▪ El usuario a modificar debe existir
Secuencia normal	1. El administrador selecciona el usuario que desea modificar 2. Actualiza los datos pertinentes 3. Confirma la operación
Postcondición	Los datos del usuario quedan actualizados
Excepciones	E1. El usuario no existe E2. El formato de los datos introducidos es incorrecto
Comentarios	

Tabla 3.13: CU13 - Modificar usuario

CU - 14	**Eliminar usuario**
Objetivos asociados	Eliminar una cuenta de usuario del sistema
Entradas	▪ Email (ID) actual del usuario a eliminar

Salidas	Mensaje de confirmación
Precondición	▪ Necesario estar identificado como **administrador** ▪ El usuario a eliminar debe existir
Secuencia normal	1. El administrador selecciona el usuario que desea eliminar 2. Confirma la operación
Postcondición	El usuario y sus modelos asociados son eliminados del sistema
Excepciones	E1. El usuario no existe E2. El usuario que se intenta eliminar es administrador único
Comentarios	

Tabla 3.14: CU14 - Eliminar usuario

.

Capítulo 4

Tecnologías empleadas

La tecnología se alimenta a sí misma. La tecnología hace posible más tecnología.

Alvin Toffler

4.1. Introducción

En el presente capítulo se proporciona una breve descripción sobre las tecnologías empleadas en la programación de este trabajo.

4.2. Tecnologías empleadas en la totalidad de la aplicación

A continuación se exponen las tecnologías utilizadas tanto en el lado del cliente como en el lado del servidor de la aplicación.

4.2.1. HTML5

HTML es un lenguaje de marcado de hipertexto (*HyperText Markup Language* en inglés). Es el estándar para la definición del esquema de páginas web [6]. Se ha empleado su última revisión, HTML5, que incopora las últimas novedades a la vez que incorpora mejoras de compatibilidad para distintos navegadores.

En este trabajo actúa como recurso *full-stack*, es decir, interactúan con el contenido HTML tanto el usuario (desde el cliente), como el servidor para generar contenido dinámico dentro de un esquema HTML fijo y procesar entradas del usuario.

4.3. Tecnologías en el lado del cliente

En esta sección se presentan las herramientas empleadas para desarrollar el lado cliente de la aplicación.

4.3.1. Bootstrap 4

Bootstrap es uno de los *frameworks* más populares para el diseño *front-end*. De hecho, se autodenomina «la librería más popular de HTML, CSS y JS del mundo» [7]. Fue creado por los desarrolladores de Twitter en 2011.

Se caracteriza por el *responsive design*, esto es, se adapta automáticamente al tamaño de pantalla de dispositivos muy dispares. Posee estilos predefinidos, así como una gran variedad de *plugins* en JavaScript que ayudan a conseguir una web más interactiva. También es conocido por la flexibilidad de diseño que otorga su *Grid Layout*, que permite organizar los elementos de una página en filas y columnas.

En la actualidad, es un proyecto de software libre mantenido en GitHub.[1]

4.3.2. CSS3

Las hojas de estilo en cascada (CSS por sus iniciales en inglés) son el mecanismo principal para aplicar estilos visuales a una página web. Su objetivo es vinclular unas ciertas propiedades visuales a distintos elementos HTML

Al decir que las hojas de estilo se aplican en cascada, se quiere decir que si un elemento HTML no tiene ningún estilo asociado, heredará los estilos de su elemento padre. Por el contrario, si el elemento en cuestión tiene aplicado un estilo que difiere del de su padre, prevalece el suyo propio [8].

Los estilos se aplican con selectores, que pueden afectar a todos las apariciones de una misma etiqueta, a todos los elementos de una determinada clase o bien a un único elemento dado por su identificador.

CSS es un estándar mantenido por el World Wide Web Consortium (W3C).

4.3.3. JavaScript

JavaScript es un lenguaje de programación interpretado basado en el estándar ECMAScript[2]. Se incorpora en todos los navegadores modernos desde

[1] `https://github.com/twbs/bootstrap`

[2] Para más información véase `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources`

2012 [9]. Permite la ejecución de código en el lado del cliente, dotando a los sitios web de una mayor interacción con el usuario [10].

Hace uso del árbol DOM[3] para mapear los elementos HTML, de forma que se pueden utilizar distintos selectores de nodos (elementos) del árbol y modificarlos o examinar su contenido.

Al tener todo el contenido HTML ordenado de esta forma, JavaScript también permite añadir o eliminar elementos de forma dinámica.

4.3.4. JQuery

JQuery es uno de los *frameworks* más populares para JavaScript. Su utilidad radica en la simplificación de complejas sentencias de código JavaScript mediante el uso de selectores.

Es ampliamente conocido por la facilidad que proporciona para manipular el árbol DOM, gestionar eventos y realizar peticiones AJAX [11].

El proyecto JQuery es software libre bajo licencia MIT, se desarrolla desde 2006 y en 2019 era usado en un 86 % de los sitios web a nivel mundial [12].

4.3.5. AJAX

AJAX (acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*) no es una tecnología en sí misma, sino más bien un recuerso que permite establecer un canal de comunicación entre el cliente y el servidor mediante el uso de otras tecnologías existentes. En concreto, mediante el uso de XHR (*XML HTTP Requests*).

Frente a las aplicaciones web tradicionales, al hacer peticiones asíncronas, AJAX permite que los sitios web sean más rápidos y dinámicos, procesando en el servidor datos de usuario sin tener que recargar la página [13].

Las peticiones AJAX, al basarse en HTTP, pueden ser de tipo `GET` o `POST`. Aceptan como paramétros una página de destino (que es, en esencia, el *script* que atenderá la petición en el servidor), datos codificados en formato XML o JSON[4] y otras opciones (como funciones de tratamiento del código HTTP devuelto por la petición).

[3] El árbol DOM (*Document Object Model*) es una estructura que almacena todos los elementos HTML de una página web de forma ordenada.

[4] JSON (*JavaScript Object Notation*) es un formato para representar objetos JavaScript como texto plano.

AJAX puede utilizarse mediante código JavaScript nativo, o mediante algún framework JavaScript, como JQuery (lo que generalmente es más intuitivo).

4.3.6. Cytoscape JS

Cytoscape JS es una librería de código abierto para grafos escrita en JavaScript. Contiene multitud de funciones útiles en teoría de grafos [14]. Sus funcionalidades se reparten en tres grupos: las relativas al grafo como entidad, las relativas a un conjunto de elementos del mismo y las que aplican a un único nodo (un vértice y sus relaciones).

Permite definir un grafo en la inicialización del objeto, con unos vértices y aristas predeterminadas, e ir modificándolo con las funciones correspondientes. También ofrece la posibilidad de crear un grafo desde cero, sobre un objeto grafo inicializado a vacío.

También es importante destacar que esta librería permite establecer relaciones de parentesco entre nodos, factor que será clave en la implementación de este trabjo. Además, Cytoscape ofrece otras opciones de configuración, tales como el diseño o el *layout* del grafo, animaciones o respuesta a eventos.

El núcleo de Cytoscape JS está mantenido en GitHub[5], si bien cuenta con una amplia comunidad de desarrolladores que crea extensiones propias con propósitos muy variados. De hecho, en el desarrollo de este trabajo se han utilizado las siguientes:

- Cytoscape No-Overlap[6]: se trata de una extensión que evita que se solapen los nodos del grafo, ya que en el caso de este trabajo generaría una situación de gran incomodidad para el usuario.

- Cytoscape CoSE Bilkent[7]: se trata de un layout que además proporciona funciones adicionales como animaciones en el diseño y establecer un orden para los nodos, entre otras. [15]

4.4. Tecnologías en el lado del servidor

A continuación se detallan las tecnologías empleadas en el desarrollo de la aplicación en el lado del servidor.

[5] `https://github.com/cytoscape/cytoscape.js`
[6] `https://github.com/mo0om/cytoscape-no-overlap`
[7] `https://github.com/cytoscape/cytoscape.js-cose-bilkent`

4.4.1. Apache HTTP Server

Apache es un servidor web impulsado desde 1995 por la Apache Software Foundation. Es el software de servidor más utilizado a nivel mundial desde 1996 [16].

Debe su popularidad a su robustez y su gran capacidad de personalización según las necesidades requeridas. Está disponible para Windows, Mac OS, Linux y otros sistemas operativos.

4.4.2. MongoDB Community Server

Como se introduce en el capítulo 2 de esta memoria, MongoDB es una de las plataformas más populares para crear bases de datos NoSQL orientadas a documento.

Community Server[8] es la versión gratuita de MongoDB, que se compone de un servicio de servidor (el demonio mongod) y una *shell* de cliente que permite tanto administrar la base de datos como lanzar consultas a la misma. Como curiosidad frente a las que ofrecen otras plataformas, cabe destacar que la *shell* de MongoDB permite ejecutar código JavaScript, lo cuál simplifica la labor de insertar cantidades masivas de datos mediante *scripting*.

Para el funcionamiento de la aplicación desarrollada, será necesario instalar MongoDB Community Server en el mismo *host* que está ejecutando el servidor HTTP de Apache.

4.4.3. PHP 7

PHP (acrónimo recursivo de PHP Hypertext Preprocessor) es un lenguaje de *scripting* de propósito general [17]. En concreto, es uno de los lenguajes más utilizados para la programación del lado del servidor de aplicaciones web. Su gran popularidad es debida a la facilidad para incrustar código PHP en documentos HTML y generar contenido de forma dinámica sobre los mismos.

Para ejecutar código PHP en un servidor es necesario tener instalado el intérprete del lenguaje, generalmente en forma de demonio o servicio del sistema. En este trabajo se utiliza PHP para procesar datos introducidos por el usuario e interactuar con la base de datos de la aplicación.

Para más información se puede consultar el sitio web del proyecto PHP.[9]

[8] `https://www.mongodb.com/try/download/community`
[9] `https://www.php.net/`

4.4.4. PHP Mailer

PHP Mailer es una librería de código abierto que permite enviar correos electrónicos desde un servidor que ejecuta PHP. Su uso es relativamente sencillo, ya que solo es necesario configurar una cuenta de correo introduciendo sus parámetros y confeccionar el correo a enviar (tanto el cuerpo del mensaje como su cabecera, incluyendo destinatarios, asunto y otras opciones).

En realidad, el propio motor de PHP incorpora código nativo para enviar emails desde el servidor. Sin embargo, es muy limitado porque no cubre casos de uso bastante habituales, como por ejemplo adjuntar archivos a un email. En la situación de desarrollo actual de este trabajo se podría haber escogido esta solución; sin embargo, se ha preferido utilizar PHP Mailer de cara a facilitar la escalabilidad de la aplicación en un futuro.

El proyecto se encuentra mantenido en GitHub[10].

4.5. Otros recursos empleados

En el siguiente apartado se engloban otros recursos tecnológicos que se han empleado durante el desarrollo del proyecto, aunque no sean estrictamente necesarios para el funcionamiento de la aplicación.

4.5.1. GitHub

GitHub es una de las múltiples herramientas para el control de versiones de código basadas en el sistema Git. Como la mayoría de las plataformas implementadas en base a Git, GitHub permite crear distintos repositorios en los que, mediante *commits*[11], se pueden ir almacenando diferentes versiones del código de un software.

Además permite crear ramas para facilitar el trabajo en paralelo, pudiendo unir el código de una rama secundaria a la rama principal cuando está listo o ha sido revisado. GitHub ofrece un sinfín de funcionalidades adicionales, como permitir a la comunidad de desarrolladores enviar modificaciones de código documentadas (conocidas como *pull requests*).

En el marco de desarrollo de este trabajo, además de utilizarse como sistema de control de versiones, GitHub ha sido la fuente principal de librerías y tecnologías desarrolladas por terceros, como se viene documentando en este capítulo.

[10] `https://github.com/PHPMailer/PHPMailer`

[11] Un *commit* es un conjunto de cambios confirmados y almacenados, en el caso de GitHub, con una pequeña descripción sobre los cambios realizados y su fecha.

Capítulo 5

Modelo de datos

Datos, datos, datos. No puedo hacer ladrillos sin arcilla.

Sherlock Holmes

5.1. Introducción

En este capítulo se define el modelo de persistencia de la aplicación, abarcando desde los motivos para la elección de la plataforma hasta la integración con el resto de la herramienta.

5.2. Modelo E-R

Para diseñar el modelo de datos de la aplicación se ha utilizado un diagrama entidad-relación. En el diagrama aparecen tres entidades:

- **Usuario**: la entidad usuario representa la información almacenada sobre un usuario registrado. Sus atributos incluyen el identificador único generado por MongoDB, el nombre y los apellidos del usuario, el email con el que se ha registrado y la contraseña establecida.
- **Modelo**: simboliza la información que se almacena sobre los modelos creados por un usuario. En concreto, para cada modelo se almacena un identificador único, su nombre, la fecha de última modificación, la representación gráfica del modelo (objeto Cytoscape) y los documentos insertados. Además, también almacena el email del propietario del modelo, atributo que toma a través de la relación «crea».
- **PasswordRecovery**: esta entidad estructurará la información generada cuando un usuario solicita recuperar su contraseña. En particular, contiene un identificador de MongoDB, un token único y la fecha de

emisión del token. También se almacena el email del usuario que efectúa la solicitud, obtenido a través de la relación «solicita».

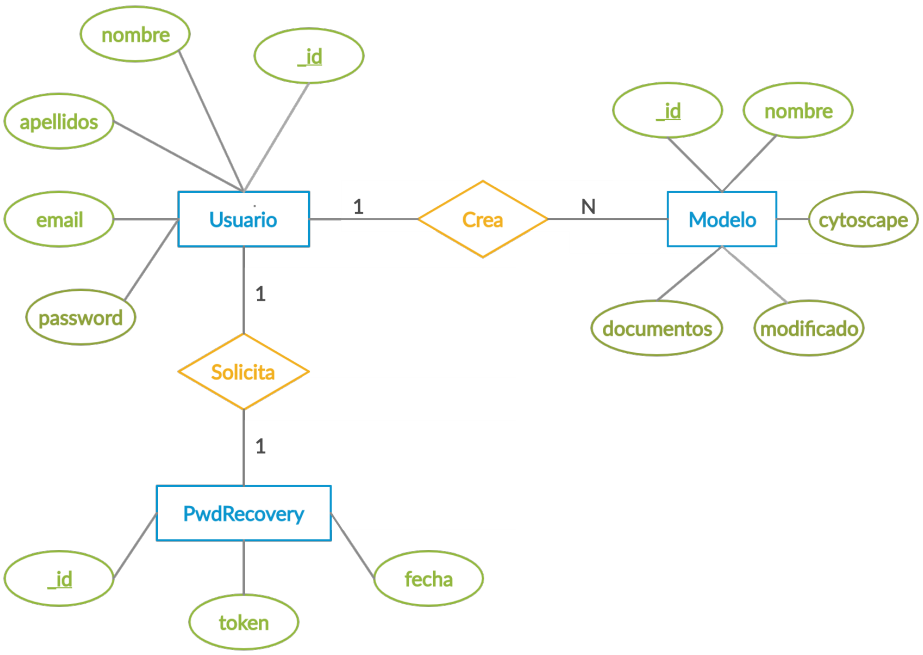

Figura 5.1: Diagrama entidad-relación

5.3. Implementación

Para la implementación del modelo de datos de este trabajo, se ha optado por una base de datos NoSQL orientada a documento, concretamente en MongoDB.

Se ha escogido esta plataforma por las facilidades que ofrece, tanto para almacenar información con formatos no uniformes, como para su integración con lenguajes del lado del servidor.

En las siguientes secciones de este capítulo, se cuenta con detalle la estructura del modelo empleado.

5.4. Estructura de la base de datos de la aplicación

Al particularizar el modelo entidad-relación en un modelo para MongoDB, resulta una base de datos estructurada en tres colecciones: usuarios,

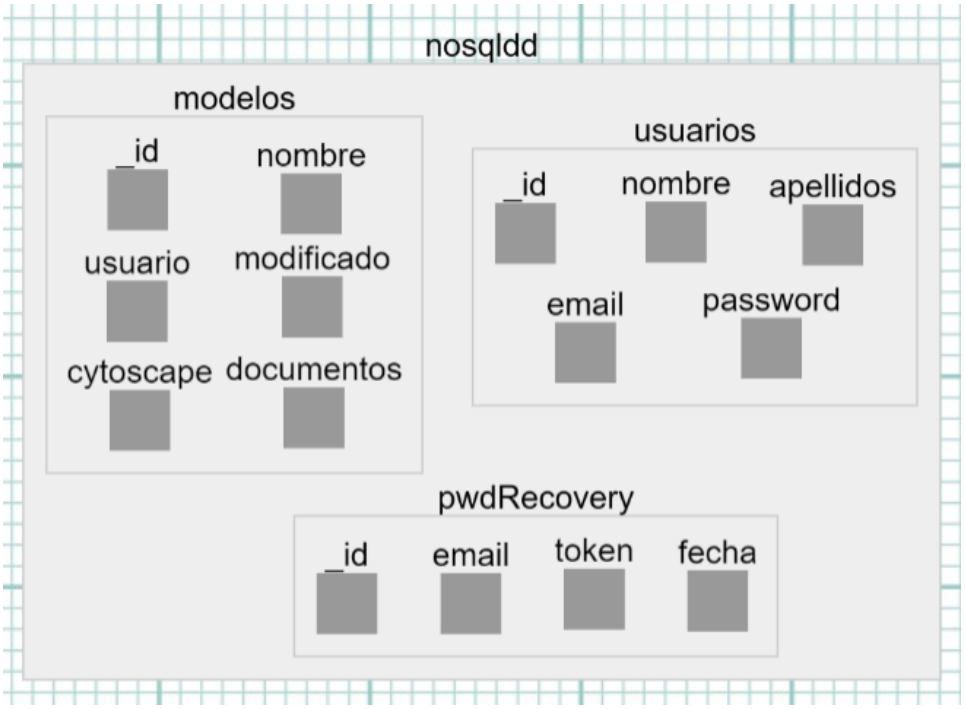

Figura 5.2: Representación gráfica del modelo de datos

modelos y pwdRecovery. Para ilustrar la lectura de las siguientes secciones, en la figura 5.2 se presenta una representación gráfica más concreta del modelo de datos de la aplicación.

5.4.1. Colección usuarios

La colección usuarios almacena la información de registro de los usuarios de la aplicación. En concreto:

- _id: el objeto ObjectId() generado automáticamente por MongoDB. También podría haberse establecido como id el email del usuario, ya que será un campo único para cada documento. Esta condición se verifica al crear un usuario desde el código PHP (si ya existe un usuario con el email introducido, se cancela la operación).
- nombre: el nombre de pila del usuario.
- apellidos: los apellidos del usuario.
- email: el correo electrónico con el que se registra el usuario. Como se indica anteriormente, los valores de este campo son únicos.
- password: almacena el *hash* de la contraseña elegida por el usuario. Se genera con la función `password_hash(string $pwd)` de PHP, que

utiliza por defecto el algoritmo bcrypt (basado en el cifrado Blowfish) [18].

5.4.2. Colección modelos

En la colección modelos se almacena la información de los modelos que ha creado un usuario. En particular:

- _id: el objeto ObjectId() generado automáticamente por MongoDB. Se utiliza como identificador del modelo, ya que solo este campo tiene valores únicos.
- usuario: almacena la dirección de correo electrónico del usuario que ha creado el modelo en cuestión. Servirá para buscar los modelos de un usuario.
- nombre: cadena de caracteres que representa el título del modelo
- cytoscape: representación JSON del objeto Cytoscape de cada modelo. Contiene la información necesaria para representar la estructura del modelo creado.
- documentos: array que contiene los documentos (datos) introducidos en el modelo.
- modificado: campo de tipo ISODate que almacena, en milisegundos, el tiempo transcurrido desde el *epoch*[1] hasta la fecha UTC[2] en la que se guardaron cambios sobre el modelo por última vez.

5.4.3. Colección pwdRecovery

La colección pwdRecovery es fundamental para el servicio de recuperación de contraseña. En ella, se asocia el email del usuario que ha solicitado una nueva contraseña a un token único. Sus documentos incluyen los siguientes campos:

- _id: identificador generado automáticamente por MongoDB. También podría establecerse como identificador del documento el propio token generado, que de igual manera es único.
- token: token único generado al solicitar una nueva contraseña. Se genera al calcular la función resumen (con el algoritmo SHA-256) de la concatenación del email del usuario, su contraseña antigua y el instante de tiempo de la petición. Expresado en forma de pseudocódigo:

[1] El *epoch* es la fecha en la cual establecen el origen de los tiempos los sistemas UNIX, en general acordada como el 1 de enero de 1970.

[2] UTC es el estándar principal de medición de tiempo.

```
token = hash("sha256", email + password + tiempoActual)
```

De esta forma se garantiza que cada token emitido es único.

- email: la dirección de correo electrónico del usuario que ha solicitado el cambio de contraseña asociado a ese token.
- fecha: campo de tipo ISODate que almacena la fecha exacta de emisión del token.

Por motivos de seguridad, el token generado caduca 24 horas después de su emisión. Para cubrir esta funcionalidad, se ha creado un índice TTL[3] con duración de 86400 segundos (24 horas) sobre la clave fecha de cada documento [19].

[3] *Time To Live*, unidades de tiempo transcurridas las cuales un valor almacenado expira.

Capítulo 6

Arquitectura de la aplicación

Programar sin una arquitectura o diseño en mente es como explorar una gruta sólo con una linterna: no sabes dónde estás, dónde has estado ni hacia dónde vas.

Danny Thorpe

6.1. Introducción

La implementación de este Trabajo de Fin de Grado se ha realizado en forma de aplicación web. En este capítulo se describe la arquitectura de dicha aplicación, incidiendo especialmente en las tareas ejecutadas por cada parte y cómo se relacionan las partes entre sí.

6.2. Tipo de arquitectura

Al tratarse de una aplicación web, la arquitectura de la implementación es de tipo cliente-servidor. Esto quiere decir que el código de la aplicación está alojado en un servidor web y, cuando un cliente accede a dicho *host*, se le suministra el código del lado cliente con el que puede interactuar el usuario.

A continuación se desglosan las tareas a ejecutar para el correcto funcionamiento de la aplicación, categorizándolas en función de la parte de la arquitectura responsable de las mismas.

6.3. Tareas realizadas en el servidor

En el servidor que alberga la aplicación se realizan las siguientes tareas:

- **Servir el contenido de la aplicación**: en esencia, atender las peticiones HTTP que llegan al servidor y contestar con la página solicitada o el error correspondiente.
- **Procesar las entradas de usuario**: incluye filtrar las entradas del usuario (escapando caracteres especiales, por ejemplo) y validar que la entrada se ajuste a unos requisitos establecidos (por ejemplo, verificar la longitud de una cadena introducida).
- **Generar contenido dinámico**: en función de entradas de usuario y otra información variable (por ejemplo, la fecha actual), generar contenido dinámico para incrustarlo dentro del contenido estático de las páginas HTML.
- **Intercambiar información con la base de datos**: con las entradas de usuario validadas, generar consultas para insertar y recuperar información de la base de datos. El servidor de la base de datos puede estar en el mismo host que el servidor HTTP o no.

Como se puede observar, una de las tareas que debe realizar el código del lado del servidor es comunicarse con la base de datos. Tal como se explicaba en el capítulo 4, el lenguaje de programación empleado en el lado del servidor es PHP. Por tanto, es necesario conectar el motor de PHP con la base de datos para lanzarle consultas.

PHP facilita un paquete de extensiones que implementa estas tareas. Se trata del paquete MongoDB Driver, que si bien implementa todas las operaciones necesarias, lo hace a bajo nivel. Esto puede complicar la programación de las consultas a la base de datos, por lo que desde el propio manual de PHP recomiendan llamar a estos métodos a través de una API de más alto nivel [20].

En el propio repositorio GitHub de MongoDB[1] se facilita la implementación de una API de alto nivel. Sin embargo, para el desarrollo de este trabajo se ha preferido desarrollar una API propia. El motivo de esta elección ha sido simplificar la lectura del código, obviando procedimientos que no son necesarios para esta aplicación.

En la clase `MongoDatabase.php` del código se puede consultar la implementación de la citada API.

6.4. Tareas realizadas en el cliente

La parte cliente es la encargada de realizar los siguientes cometidos:

[1] `https://github.com/mongodb/mongo-php-library`

- **Enviar peticiones al servidor**: el navegador puede enviar peticiones HTTP de tipo `GET` para solicitar una página y de tipo `POST` para enviar esa misma página con datos introducidos por el usuario. El cliente puede enviar otro tipo de peticiones. En particular, en esta aplicación también se utilizará XHR para encapsular peticiones AJAX y ejecutar *scripts* concretos del servidor sin recargar la página.

- **Mostrar las páginas HTML recibidas**: el cliente es el encargado de interpretar los documentos HTML (y sus estilos, si los tienen) recibidos de las peticiones HTTP `GET` y mostrarlos en el navegador.

- **Validación de datos en el cliente**: esta tarea no es estrictamente necesaria en una arquitectura cliente-servidor, pero mejora considerablemente la experiencia de usaurio. Esto es debido a que no se permite enviar los formularios (peticiones HTTP `POST`) hasta que los datos cumplen ciertas condiciones. En el caso de esta aplicación, se utiliza la validación de HTML5 para marcar campos obligatorios y comprobar que las direcciones de correo electrónico introducidas son válidas.

- **Ejecutar código del cliente**: las peticiones HTTP `GET` no solo se usan para solicitar documentos HTML, sirven para pedir cualquier tipo de recurso. En el caso de esta implementación, también se utilizan para solicitar los *scripts* del cliente, desarrollados en JavaScript. Esta es una parte fundamental del proyecto desarrollado, ya que permite al usuario interactuar con el diseñador gráfico y crear sus modelos. En el siguiente capítulo se explican los *scripts* del cliente con mayor detalle.

Capítulo 7

Diseño de la aplicación

El diseño no es solo lo que ves, sino cómo funciona.

Steve Jobs

7.1. Introducción

Como se comenta en el capítulo anterior, la implementación del trabajo se basa en una aplicación web, ya que uno de los principales objetivos es que sea lo más portable posible. Al desarrollar una aplicación web, se garantiza que el usuario solo necesitará hacer uso de un navegador web actualizado, evitando tener que instalar la aplicación y librerías o código de terceros (por ejemplo, Java).

La aplicación se estructura por tanto como un conjunto de páginas HTML, con scripts PHP y JavaScript que permiten al cliente y al servidor interactuar con las mismas.

En este capítulo se explican los objetivos de las diferentes páginas y cómo se relacionan con los scripts y el resto de tecnologías expuestas en el capítulo 4. Además, se comentarán los detalles más relevantes de la implementación. A modo de referencia y para facilitar la comprensión de este capítulo, en la figura 7.1 se muestra un diagrama de flujo que ilustra las secuencias de uso principales.

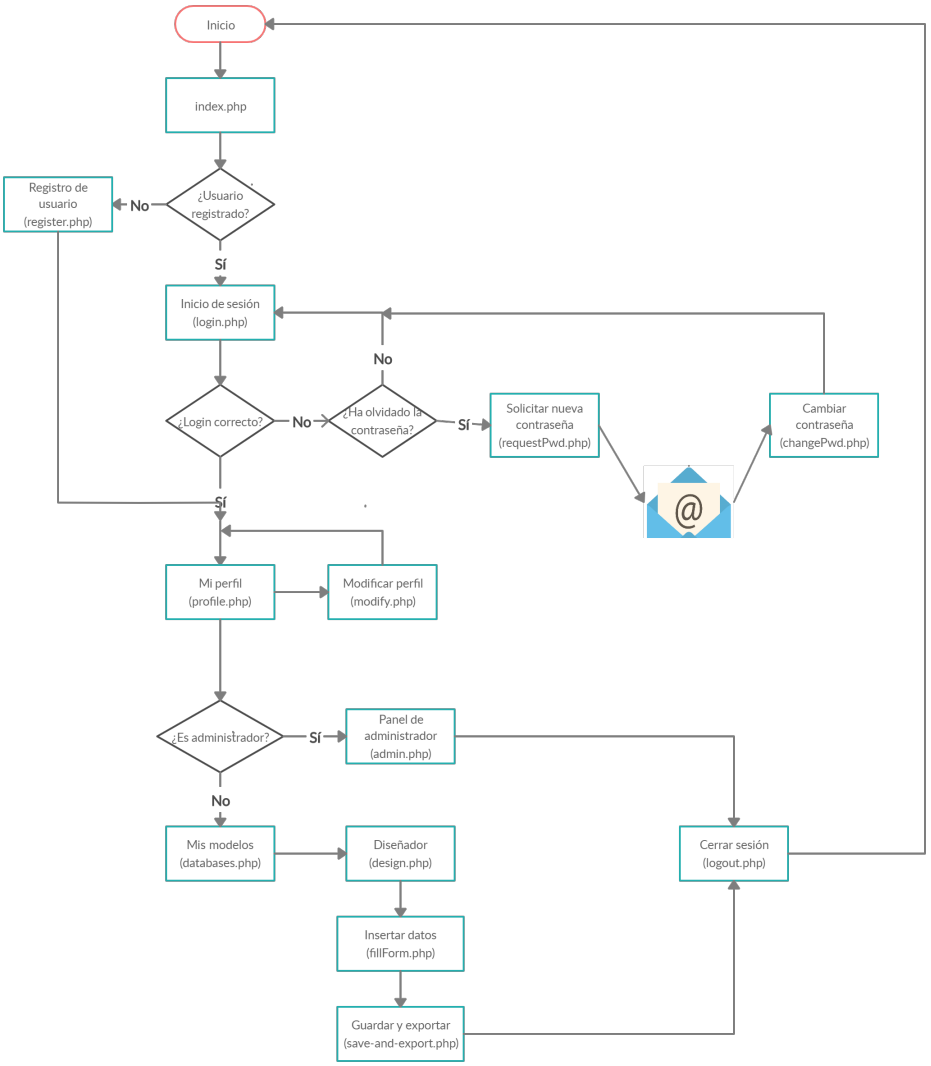

Figura 7.1: Diagrama de flujo ilustrando las secuencias de uso principales

7.2. Página de inicio

La página de inicio de la aplicación (`index.php`) es la que visualiza el usuario cuando llega al sitio web y aún no se ha identificado. En esta página simplemente se muestra un carrousel con imágenes de la aplicación, presentando brevemente todas las características de la aplicación.

El objetivo es hacer la aplicación atractiva al usuario e invitarlo a registrarse. Por tanto, una vez que el usuario se registra e inicia sesión, deja de ser necesaria. En consecuencia, si el usuario está identificado esta página

deja de ser accesible desde la barra de navegación. Si aun así se intentara acceder manualmente ingresando su URL, el servidor redirigirá al usuario a su listado de modelos, apartado cuyo funcionamiento se detalla más tarde en este capítulo.

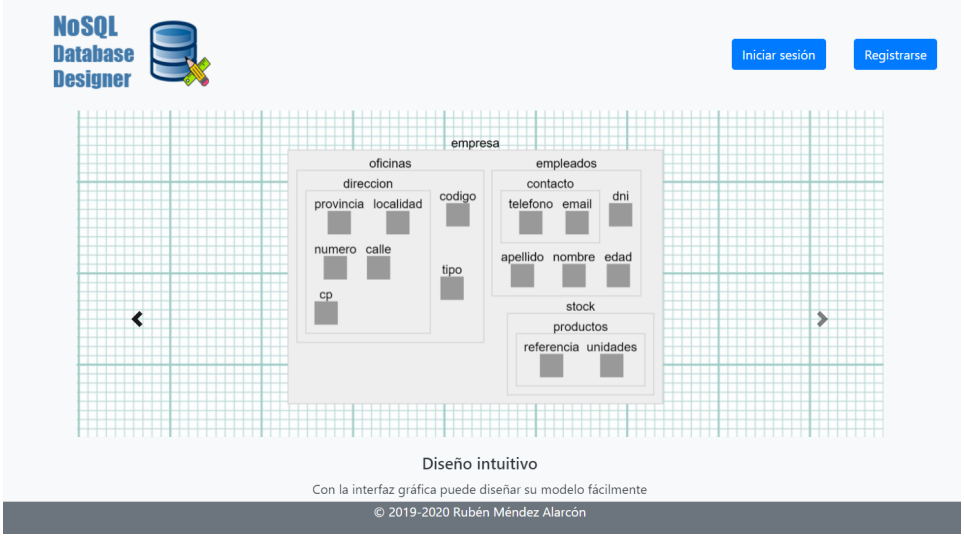

Figura 7.2: Vista de la página de inicio

Tanto desde la página de inicio como desde el resto se tiene acceso a una barra de navegación, que permite el acceso a todas las funciones de la aplicación.

7.3. Registro de usuarios

Dado que para usar la aplicación se exige a los usuarios estar identificados, una parte primordial de la misma es el registro de usuarios.
En este caso, el registro de usuario consta de un formulario HTML en el que se solicitan datos personales (nombre, apellidos, correo electrónico y una contraseña).

Al pulsar «Enviar» se envía el formulario al servidor, donde es procesado. Allí se validan los datos del formulario por motivos de seguridad (el usuario podría haber bloqueado la ejecución de código JavaScript, por ejemplo). Si la validación es correcta, se inserta el documento correspondiente al registro en la colección usuarios de la base de datos. Acto seguido, la sesión queda iniciada automáticamente.

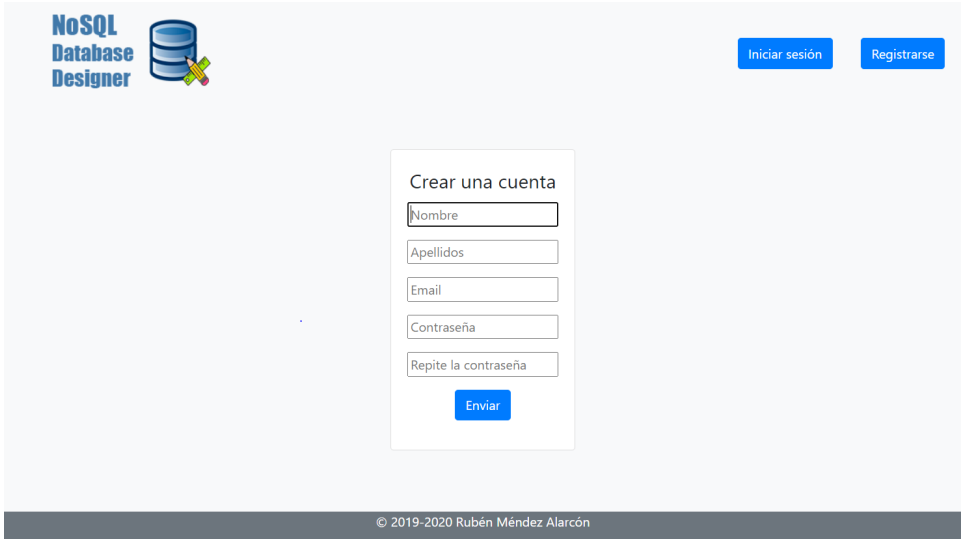

Figura 7.3: Vista de la página de registro de usuarios

7.4. Login o acceso a la aplicación

La página de login o inicio de sesión permite al usuario identificarse en la aplicación y hacer uso de todas sus características.

El núcleo de su lógica figura en un formulario que recoge el email y la contraseña del usuario. Al enviarse el formulario al servidor, se validan los datos. En primer lugar, se comprueba que el email está registrado y, si es así, se genera el *hash* de la contraseña introducida y se contrasta con el almacenado en la base de datos.

Si las credenciales son correctas, se inicia la sesión del usuario y se le redirige a su listado de modelos. En caso contrario, se notifica el error y se le sugiere la opción de recuperar su contraseña.

7.5. Recuperación de contraseña

Como se indica en el apartado anterior, una vez que el usuario hace un intento erróneo de login se le sugiere recuperar su contraseña.

Para cubrir esta funcionalidad, al hacer click en el enlace se redirige al usuario a la página `requestPwd.php`, donde encontrará un formulario simple

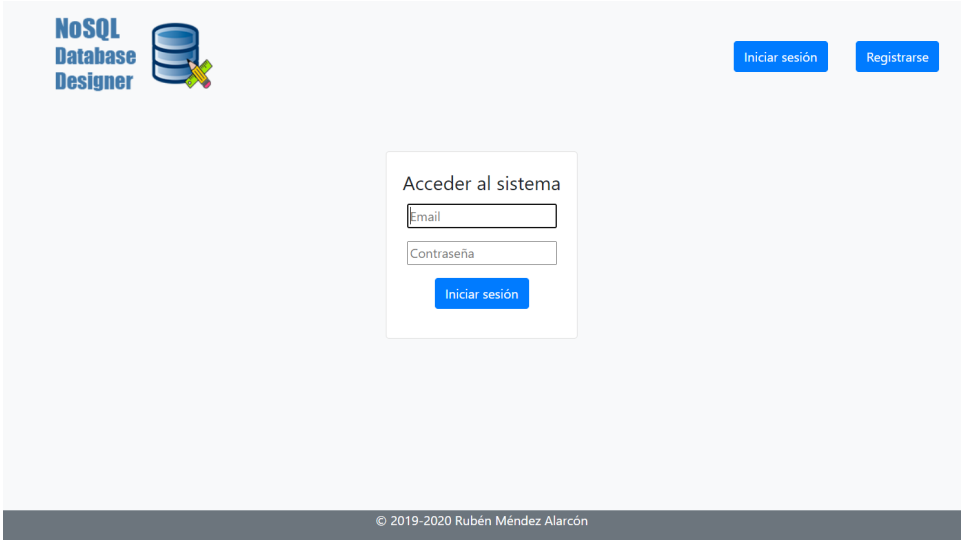

Figura 7.4: Vista de la página de inicio de sesión

para introducir su dirección de correo electrónico. Para enviar el formulario, es condición necesaria que la entrada del usuario sea una dirección de correo electrónico válida. Esta condición se comprueba también en el servidor al procesar el formulario.

Si la dirección de correo electrónico está registrada, se genera un *token* que se inserta en la base de datos. Este proceso ya se explicó en la sección 5.4.3 esta memoria. A continuación, mediante una instancia de la clase `PHPMailer.php` se envía un *link* con el *token* generado a la dirección de correo indicada.

Finalmente, se muestra un mensaje de confirmación. Por motivos de seguridad, se ha elegido que el mensaje de confirmación se muestre siempre que el email sea válido, aunque no esté registrado. El objetivo es que un usuario con fines maliciosos no pueda determinar si la dirección de un tercero está registrada o no.

Cuando el usuario recibe el email y hace click en el vínculo, es enlazado a la página `changePwd.php`. En esta página se muestra un formulario con tres campos: uno para introducir el email nuevamente (por razones de seguridad) y otros dos para escribir y confirmar la nueva contraseña. Si los datos pasan la validación en el servidor, se actualiza en la base de datos el documento correspondiente al usuario con la nueva contraseña y se muestra un mensaje de confirmación. En otro caso, se muestra el error acontecido.

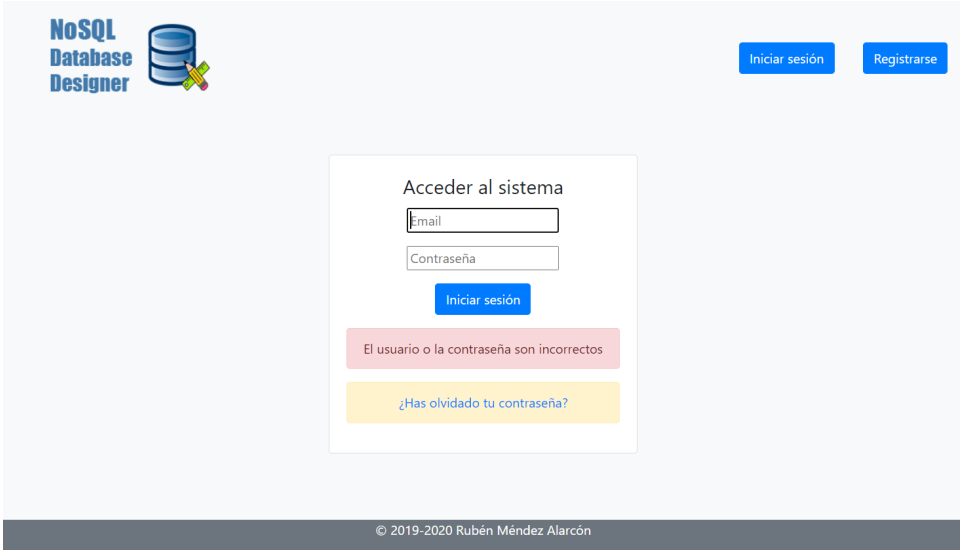

Figura 7.5: Vista de la página de inicio de sesión tras un intento de login incorrecto

7.6. Mis modelos

Desde este módulo de la aplicación, el usuario puede crear un nuevo modelo o bien consultar los que ya ha creado. Si opta por crear un nuevo modelo, al pulsar el botón correspondiente se abre un *prompt* en el que se pide un nombre. Si el nombre es válido (no es una cadena vacía y no contiene espacios ni caracteres especiales) se crea el nuevo modelo. En otro caso, se informa de que el nombre no es válido y se cancela la operación.

Al crearse el nuevo modelo, se inicializa una variable de sesión en el cliente que indica que ese modelo está abierto actualmente, y se redirige al usuario a la página del diseñador. Si el usuario vuelve a su listado de modelos podrá ver que, además de sus modelos guardados, tiene un modelo con cambios sin guardar. Sobre este modelo puede ejecutar las siguientes opciones:

- **Descartar los cambios:** cierra el modelo sin guardar los cambios, previa confirmación. Si se acababa de crear el modelo (aún no se había guardado), se eliminará todo el modelo. Por el contrario, si se estaba editando un modelo creado en otro momento (ya estaba guardado), solo se descartan los cambios hechos en la sesión actual. Para llevar a cabo esta acción, tan solo es necesario borrar la citada variable de sesión.
- **Guardar y cerrar:** guarda los últimos cambios realizados sobre el modelo y lo cierra para poder editar otro. La tarea se realiza enviando

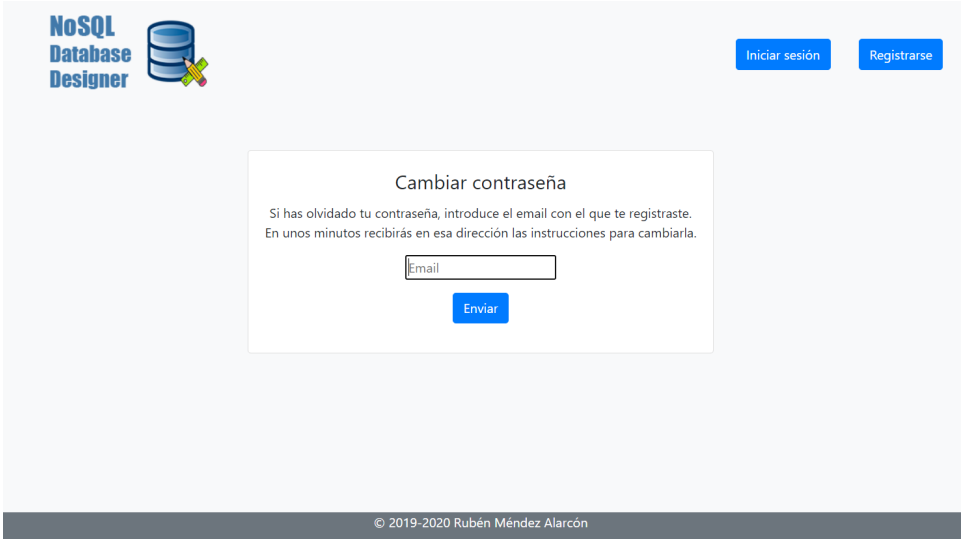

Figura 7.6: Vista de la página de solicitud de nueva contraseña

las modificaciones al servidor mediante una petición AJAX y, una vez devuelve un código de éxito (se ha insertado correctamente en la base de datos), se descarta la variable de sesión.

Nótese que cuando se habla de guardar cambios es referido a guardarlos de forma persistente en la base de datos de la aplicación. Cada vez que el usuario realiza un cambio en el diseñador se guarda en la sesión de usuario, para que no se pierda si el usuario cambia de página dentro del sitio. Sin embargo, si el usuario cierra el navegador sin hacer un guardado persistente, perderá todos sus cambios.

Por otro lado, sobre los modelos guardados se pueden ejecutar las siguientes acciones:

- **Cambiar nombre:** permite cambiar el nombre asignado a un modelo. Para ello es necesario recuperar el modelo de la base de datos (mediante AJAX). Después se cambia el campo «nombre» del documento y se cambia la información correspondiente en el diseño. Para ello es necesario recorrer el objeto Cytoscape y reemplazar el nombre del nodo que agrega toda la base de datos. Una vez completadas las modificaciones, se envía de nuevo el documento al servidor mediante AJAX. Si la petición tiene éxito, se actualiza la página; en otro caso se muestra un mensaje de error.
- **Eliminar modelo:** borra el modelo y toda su información asociada de la base de datos, sin posibilidad de recuperación. Cuando el usuario lanza esta acción, en primer lugar se pide confirmación. Si la respuesta

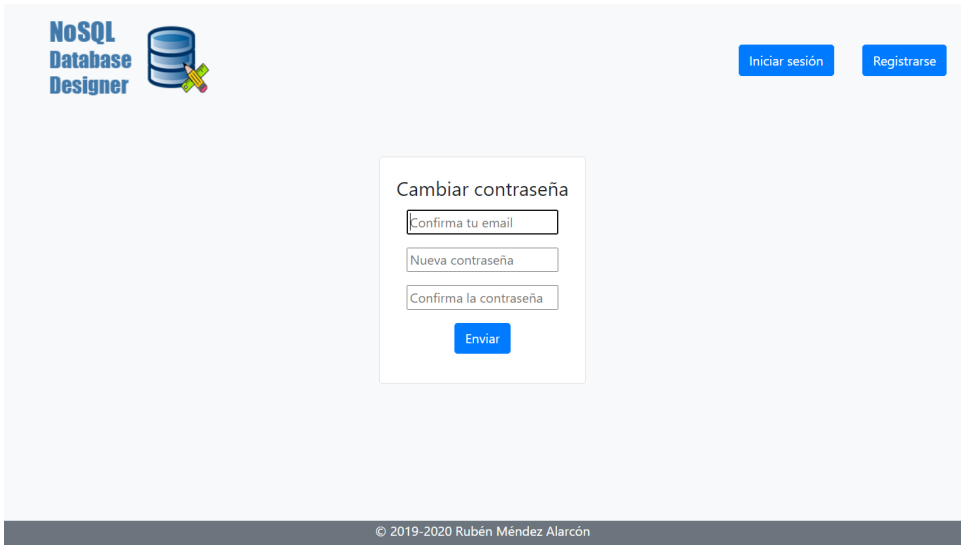

Figura 7.7: Vista del formulario para cambiar la contraseña

es afirmativa, se procede al borrado.
Para ello, se redirige al script PHP `deleteModel.php` indicando como parámetro `GET` el identificador del modelo en cuestión. El script buscará el modelo con ese identificador y, si existe, lo borrará (siempre y cuando la sesión iniciada sea del propietario del modelo). En caso contrario, no se modifica nada. Finalmente, se redirige al listado de modelos para mostrarlo actualizado.

La implementación de la eliminación de modelos también se podría hacer por medio de peticiones AJAX. Sin embargo, se ha optado por el método explicado anteriormente con el objetivo de hacer el código más legible, dado que la ejecución es lo suficientemente rápida como para no incomodar al usuario.

7.7. Diseñador

El diseñador es la piedra angular de la herramienta. Proporciona una interfaz gráfica de diseño para que el usuario pueda crear sus modelos de forma intuitiva.

Respecto a la arquitectura de su implementación, principalmente se compone de un lienzo de diseño y una botonera. El lienzo no es más que la representación gráfica del objeto Cytoscape sobre el que se sustenta toda la información del diseño; por otro lado, en la botonera se proporciona acceso a las diversas acciones que pueden ejecutarse sobre el diseño.

Figura 7.8: Vista del listado de modelos del usuario

El lienzo se imprime en pantalla acotado dentro de un marco diseñado con CSS. La representación del lienzo se hace mediante un layout estilo CoSE, proporcionado por la extensión CoSE Bilkent, como se explicaba en el capítulo 4.

Además del layout, Cytoscape permite establecer otras opciones de estilo en la inicialización del grafo. En particular, el objeto empleado tiene aplicado un estilo que muestra los nodos como cuadrados grises, y como etiqueta del nodo muestra su atributo nombre justo encima del mismo.

Como se comentaba también en el capítulo 4, Cytoscape es una librería diseñada para representar grafos mediante JavaScript. Sin embargo, la aplicación que ha tenido en este trabajo no tiene relación con los grafos. En su lugar, simplemente se han utilizado los nodos (tipo de datos que representa los vértices) para representar los agregados de datos. Por lo tanto, el objeto Cytoscape al que se hará alusión a lo largo de esta sección no tiene relaciones de un nodo a otro (aristas), sino relaciones de parentesco entre ellos.

En la pantalla del diseñador siempre aparecerá al menos un nodo: el que representa el propio modelo, con el nombre que se le haya otorgado. Dentro de este agregado se deben ir añadiendo los demás campos o agregados. Para trabajar sobre un nodo, es necesario seleccionarlo mediante un click y lanzar la acción deseada desde la botonera.

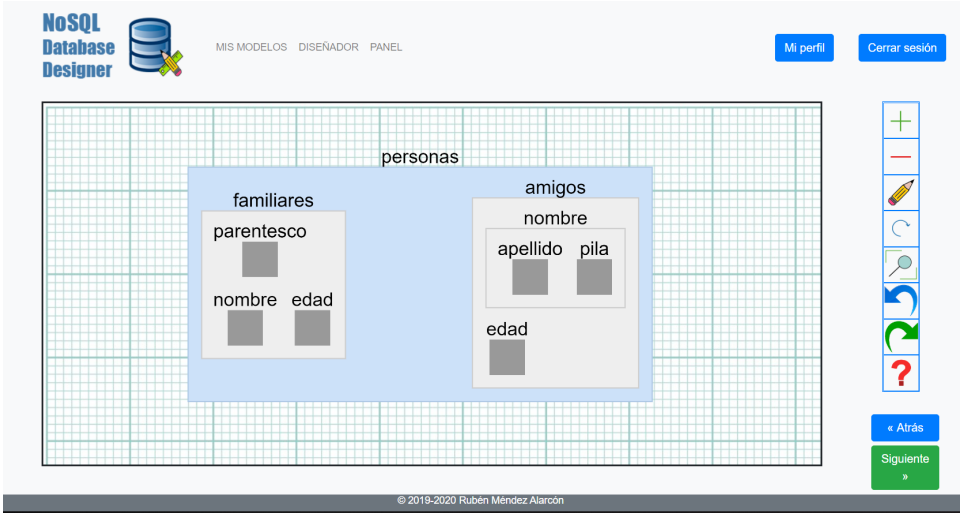

Figura 7.9: Vista del diseñador gráfico

La botonera no es más que un conglomerado de botones que permiten ejecutar las acciones disponibles sobre el modelo. Al visitar el sitio web por primera vez, se muestra un tutorial indicando cómo usar estas herramientas. Dichas acciones se enumeran a continuación, reseñando las características más destacables de su implementación.

7.7.1. Añadir item

Al pulsar este botón se abre un *prompt* que solicita el nombre del nuevo item. En primer lugar se verifica que el nombre introducido no contiene caracteres especiales y, a continuación, se determina cuál es el nodo que estaba seleccionado. Si el nodo seleccionado es el más externo del modelo (el que representa la base de datos), no se permite que el nombre contenga espacios. Esta restricción se impone para poder exportar el modelo a MongoDB, cuya sintaxis no permite espacios en los nombres de colecciones.

A continuación, se comprueba que el nodo seleccionado no tenga ya información asociada. De ser así, se destruiría la información que contiene dicho item para convertirlo en un agregado[1], por lo que se avisa al usuario y se pide su confirmación para continuar.

Finalmente, se crea un nuevo nodo cuyo padre es el seleccionado con el ratón y con nombre el introducido.

[1] Un agregado no puede contener información directamente, sino que contiene items que tienen información asociada.

7.7.2. Eliminar item

El funcionamiento es similar al de la funcionalidad anterior. En este caso, primero se comprueba si el item o agregado seleccionado tiene información asociada, en cuyo caso se advierte al usuario de que si continúa, no podrá recuperar dicha información.

Si el usuario confirma la operación, se itera por todos los documentos insertados en busca de referencias a ese item, que son eliminadas de cada documento. Finalmente, se elimina el nodo del objeto Cytoscape.

Si el item no tiene información asociada no se pide confirmación al usuario, ya que la acción podría revertirse fácilmente con el botón deshacer o volviendo a crear ese item.

7.7.3. Editar item

La herramienta editar item sirve, en esencia, para cambiar el nombre de un agregado o de un campo rellenable ya existente. Para ello, se lanza un *prompt* solicitando el nuevo nombre de la propiedad, que no puede contener caracteres especiales (ni espacios si es un nodo de primer nivel). Estas restricciones también son necesarias para la posterior generación de código para MongoDB. Finalmente, sobre el nodo que ha sido seleccionado con el ratón, se cambia el valor del atributo nombre por el introducido.

Cabe destacar que no se permite editar el nombre del nodo más externo, el que representa el modelo como concepto, ya que podría generar inconsistencias. Si el usuario intenta llevar a cabo esta acción, se le notifica que para cambiar el nombre del modelo debe hacerlo desde el listado de modelos.

7.7.4. Reiniciar lienzo

Si el usuario quisiera eliminar todo el contenido del modelo para empezar de nuevo, sería bastante tedioso tener que borrar cada agregado a mano, especialmente si el modelo es bastante extenso. Por este motivo se ha implementado el botón «reiniciar lienzo», que permite al usuario vaciar todos los items añadidos al modelo, volviendo al estado que presentaba inmediatamente tras su creación.

Para implementar esta característica simplemente se llama a la función de librería que borra un conjunto de nodos dados. En este caso, el conjunto será el que contiene todos los nodos del objeto Cytoscape. A continuación, se crea de nuevo el nodo que representará al propio modelo.

Nótese que si el modelo ya contuviera datos, se pediría una confirmación del usuario antes de reiniciar el lienzo. El botón «reiniciar lienzo» no necesita ninguna selección de ratón previa.

7.7.5. Ajustar

La librería Cytoscape permite el uso de zoom para ampliar o reducir la vista del diseño, haciendo *scroll* con el ratón. Sin embargo, no es posible ajustarlo con una gran precisión, por lo que muchas veces es deseable poder volver al estado original del zoom y presentar la vista en un ratio 1:1.

Esta función se materializa en el botón ajustar. Al accionar este botón, se llama a una función de la propia librería que «encoge» el diseño hasta adaptarlo al tamaño del lienzo donde se muetra. Este botón tampoco necesita ninguna selección de ratón previa.

7.7.6. Deshacer

Es crucial facilitar al usuario algún mecanismo que permita volver atrás tras tomar decisiones incorrectas. Para este propósito sirve el botón deshacer: permite al usuario deshacer todas las acciones realizadas, una a una.

La funcionalidad se implementa con una pila[2], en la cual se guarda el estado del objeto Cytoscape antes de ejecutar un cambio. De esta manera, deshacer un cambio es tan sencillo como desapilar la cima de la pila y cargarla en el objeto Cytoscape para que se actualice el lienzo.

En JavaScript no se implementa el tipo de datos de una pila como tal, dado que los arrays de este lenguaje están definidos a muy alto nivel y permiten operaciones similares. Aun así, para simplificar la lectura del código, se ha creado una clase que emula el tipo de datos de una pila con los nombres típicos de las operaciones: *push*, *pop*, *top*, *isEmpty* y *size*.

Hay que tratar con **especial cuidado** los cambios de nombre de un item. Esto es debido a que, aunque se almacene el estado del objeto Cytoscape antes del cambio de nombre, los objetos JavaScript son gestionados por referencias. Por tanto, al cambiar el nombre de un nodo, cambian todas las referencias al mismo, incluida la almacenada en la pila de estados anteriores.

Para solucionar esto es necesario hacer una copia en profundidad o *deep clone* del objeto, la única forma posible de obtener una copia por valor de un objeto en JavaScript [21]. En este caso, la solución pasa por apilar un

[2]Una pila es una estructura de datos que permite apilar y desapilar elementos, permitiendo consultar solo el último elemento apilado.

objeto que almacena dos parejas clave/valor: un campo `id` que almacena el identificador del nodo, y un campo `name` que contiene el nombre antes del cambio. Además, el objeto se almacena en JSON para distinguir que es un estado con un tratamiento especial al desapilarlo.

7.7.7. Rehacer

Cuando se implementa un botón de deshacer, también es conveniente tener un botón de rehacer. Esto es debido a que el usuario podría deshacer por equivocación más cambios de los que realmente necesita, necesitando rehacer cambios deshechos.

La implementación del botón rehacer también se sustenta en otra pila. En este caso, los estados apilados son los previos a pulsar el botón deshacer.

Si el usuario quisiera recuperar uno de esos cambios deshechos, al pulsar el botón rehacer se desapilaría el estado y se cargaría de nuevo en el lienzo.

7.7.8. Ayuda

Como se comenta al principio de esta sección, al visitar el diseñador por primera vez se muestra un breve tutorial que explica al usuario cómo usar las herramientas. Este tutorial se implementa con un pequeño script (`designTour.js`) que muestra en orden distintos elementos *popover* (las pequeñas notas que se muestran), a la vez que permite la navegación entre ellos.

Al resultar incómodo para el usuario que este tutorial se muestre siempre, se almacena una cookie en el cliente para que solo se muestre la primera vez que el usuario visita el sitio. Incluso en la primera visita, el usuario puede omitir el tutorial sin haberlo completado mediante un botón de cerrar.

No obstante, el usuario podría necesitar releer la ayuda más tarde, por lo que se le brinda la posibilidad de relanzar el tutorial en cualquier momento desde el botón ayuda. La implementación de este botón es tan sencilla como llamar a la función que lanza la secuencia de *popovers*.

7.8. Formulario dinámico para la inserción de datos

Una vez que el modelo está completo, para poder insertar datos en el mismo se genera un formulario dinámico desde el script `generateForm.js`, cuyos campos serán los items del modelo.

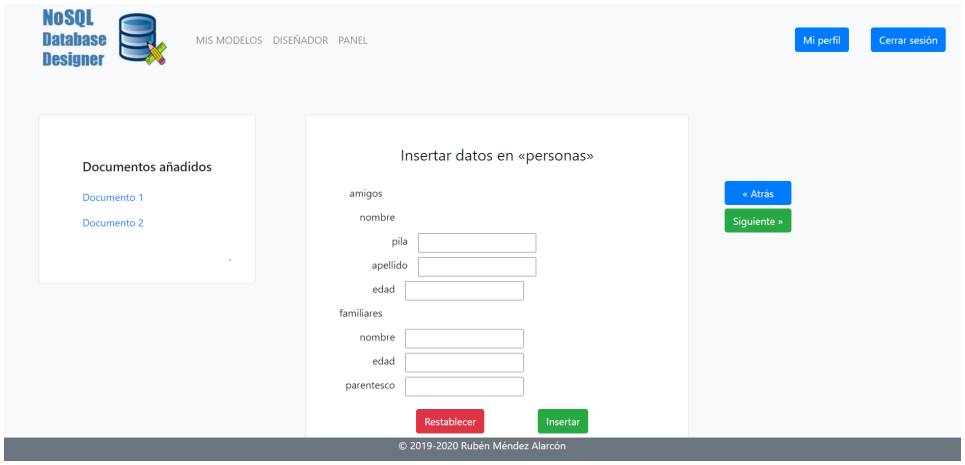

Figura 7.10: Vista del formulario para insertar datos

Antes de proceder a la explicación de la implementación, es necesario recordar al lector que los nodos del objeto Cytoscape pueden interpretarse como campos rellenables (nodos sin hijos) o agregados (nodos con al menos un hijo).

En primer lugar se imprime el nombre del nodo más externo (que por definición, es el propio modelo como conjunto). A continuación, se selecciona su primer nodo hijo. Si es un agregado, se vuelve a ejecutar el mismo algoritmo de forma recursiva. Por el contrario, si es un campo rellenable se añade un campo al formulario, cuya etiqueta será el atributo nombre del propio nodo.

Para conseguir que el formulario generado tenga un aspecto más intuitivo, se han indentado los campos según el nivel de profundidad en el que se encuentran dentro del modelo. Así, todos los campos pertenecientes a un mismo agregado aparecen alineados en otro nivel.

Una vez rellenado y enviado[3] el formulario, este es procesado (en el script `processForm.js`) para transformar la información insertada en un documento JSON, que se añadirá a un array almacenado en la sesión del cliente. De esta forma, el usuario puede ver todos los documentos insertados en cualquier momento.

Desde esta misma página se pueden ver, modificar o eliminar los documentos insertados. Para ello solo es necesario hacer click en el documento deseado. Al producirse el evento, se recupera el identificador HTML del elemento clic-

[3] Aquí se utiliza la palabra «enviado» de forma simbólica al tratarse de un formulario, pero en realidad no se envía al servidor, sino que se procesa en el cliente.

keado (que será la posición numérica del documento correspondiente en el array), se recupera el documento y se muestran sobre el formulario sus datos. Al modificar o eliminar un elemento, tan solo es necesario volver a generar el JSON de los datos (en el caso de modificaciones) y actualizar el array de documentos.

Obsérvese que al visualizar un documento ya insertado, los botones «restablecer» e «insertar» pasan a ser «eliminar» y «modificar» respectivamente, ya que son acciones diferentes.

7.9. Guardar y exportar

Al finalizar el proceso de modelado y una vez insertada la información pertinente en el modelo, el usuario puede guardar sus cambios de forma persistente y generar el código necesario para exportar su diseño a MongoDB.

Para guardar el modelo se facilita un botón a tal efecto, cuya implementación es idéntica a la del botón de guardado accesible desde el listado de modelos. Esto es, mediante una petición AJAX se envían los datos al servidor, que se encarga de insertarlos en la base de datos.

Por otro lado, para generar el código es necesario interpretar los documentos según las estructuras que implementa la base de datos de destino. En este caso, al tratarse de MongoDB será necesario crear una base de datos vacía, sobre la cual se definirán colecciones que contienen los documentos.

Por lo tanto, el generador de código crea las siguientes sentencias:

- use dbName: sentencia empleada para crear la nueva base de datos, donde dbName es el nombre dado al modelo del usuario.
- db.collection.insert(document): es necesaria una operación de inserción para cada documento creado por el usuario. Para ello, se considera la clave de nivel 0 del documento como el nombre de la colección (collection) y el resto como el propio documento (document).

Véase como ejemplo un documento insertado en el modelo «personas»:

```
{"familiares": {"nombre": "Juan", "edad": "27"}}
```

El documento anterior, interpretado en sintaxis de MongoDB, representa una colección `familiares` dentro de la cual hay un documento `{"nombre": "Juan", "edad": 27}`. En consecuencia, el código generado sería el siguiente:

```
use personas
db.familiares.insert({"nombre": "Juan", "edad": "27"})
```

Nótese que también se facilita al usuario un botón para copiar el código al portapapeles. Su implementación viene dada por el script `copyToClipboard.js`.

Figura 7.11: Vista de la página de guardar y exportar con el código generado

7.10. Mi perfil

Desde esta sección el usuario puede consultar los datos personales con los que se registró.

La lógica de esta página se sustenta en obtener el email del usuario de los datos de sesión, y con el mismo llamar a un método que devuelve todos los datos de un usuario dado su email. A continuación, los datos se imprimen en forma de ficha.

En la parte superior de la ficha de datos, el usuario puede encontrar un botón para modificar los datos del perfil.

7.11. Editar perfil

Si el usuario desea modificar sus datos personales, al hacer click en el botón correspondiente será redirigido a la página `modify.php`. En esta página

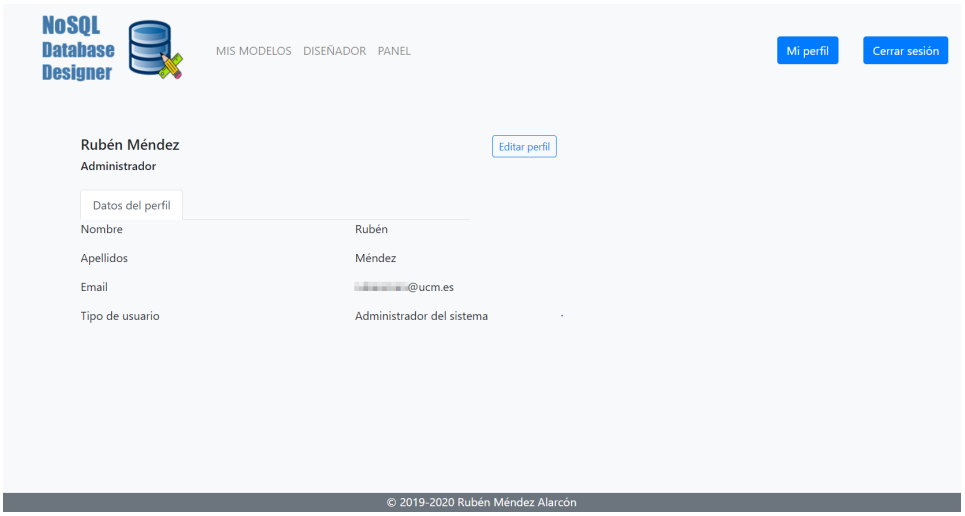

Figura 7.12: Vista de datos de un usuario en la sección «Mi perfil»

se muestra un formulario idéntico al de registro de usuarios, con la diferencia de que aparecen precargados los datos actuales del usuario. El usuario simplemente debe hacer las modificaciones que desee y confirmar los cambios. Téngase en cuenta que si se desea cambiar la contraseña, se deberá confirmar la misma para comprobar que es correcta.

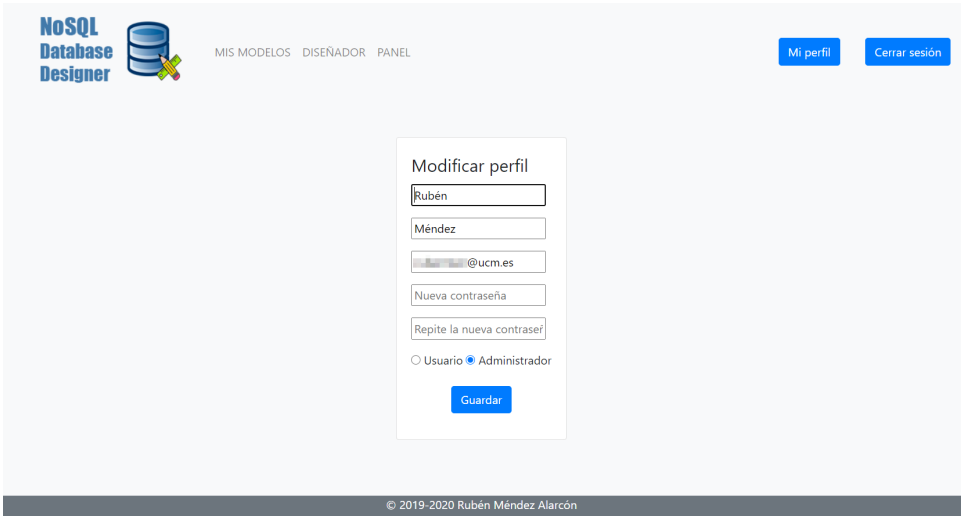

Figura 7.13: Vista del formulario de modificación de datos personales

7.12. Panel de administración de usuarios

Si el usuario identificado tiene privilegios de administrador, puede acceder a un panel desde el que puede eliminar usuarios o modificar sus datos.

La página muestra un listado llamando a un método de la clase `Usuario.php`. Dicho método se encarga de recuperar todos los documentos de la colección usuarios de la base de datos, ordenados primero por tipo de usuario y después por orden alfabético de los apellidos y el nombre.

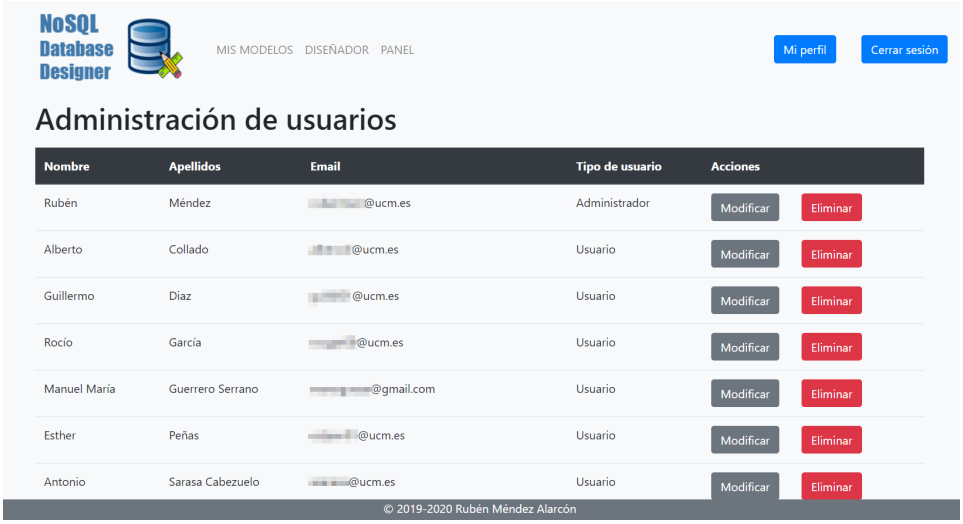

Figura 7.14: Vista del panel de administración de usuarios

Para cada usuario se muestra un botón de modificar y otro de eliminar. Al pulsar en el botón de modificar, se redirige al administrador a la página de editar perfil. Sin embargo, en el formulario no aparecerán los datos de la cuenta del administrador, sino que automáticamente se cargarán los datos del usuario a modificar. Además, al acceder al formulario siendo administrador se permite cambiar los permisos de ese usuario, añadiéndole o quitándole los privilegios de administrador. Tras enviar el formulario se redirige de nuevo al panel de administración, donde se pueden ver los datos actualizados.

Por otro lado, al hacer click en el botón eliminar (y tras confirmar la acción, ya que no se puede deshacer) se redirige al script `deleteUser.php`, pasando como parámetro `GET` el id del usuario. En el script indicado, se busca ese ID en la base de datos y se elimina si existe (comprobando siempre antes que la sesión iniciada es de administrador). A continuación, se redirige de nuevo al panel de administración y se muestra la vista con el listado ya

actualizado.

Capítulo 8

Evaluación con usuarios

Hagas lo que hagas, hazlo tan bien como para que vuelvan y además traigan a sus amigos.

Walt Disney

8.1. Introducción

En el diseño de cualquier software, es de vital importancia contar con comentarios de retroalimentación o *feedback* de los usuarios, ya que cuentan con un punto de vista diferente al del desarrollador y suele enriquecer bastante el diseño, aportando mejoras a las siguientes versiones de la aplicación.

Para valorar la usabilidad del diseño de la aplicación, se ha elaborado una encuesta en la que han participado algunos de los usuarios que han utilizado la aplicación hasta el momento de redacción de esta memoria.

En el presente capítulo se detalla el formato de la encuesta y se analizan los resultados obtenidos.

8.2. Metodología

En la evaluación con usuarios realizada han participado un total de 11 usuarios, con diferentes niveles de estudios y de conocimientos específicos en materia de bases de datos. La encuesta se compone de cuatro grupos de preguntas:

- **Datos personales**: en este bloque se recopilan datos personales del usuario, en concreto su edad, género y nivel de estudios. Las preguntas de este bloque son obligatorias. Todas proporcionan múltiples respuestas, de las cuales solo es posible escoger una opción.

- **Conocimientos de bases de datos**: para valorar los resultados finales, es necesario conocer de qué nivel de conocimientos específicos parte el usuario, por lo que se le hacen algunas preguntas relativas a su experiencia con bases de datos. Al igual que en el bloque anterior, todas las preguntas son de obligada respuesta y solo es posible seleccionar una de las opciones ofrecidas.

- **Nivel de satisfacción con la aplicación**: finalmente, una vez se ha perfilado al encuestado, se le pregunta sobre su nivel de satisfacción con el funcionamiento y el diseño de la aplicación, primero desglosado por funcionalidades y finalmente a título general. Algunas de las preguntas no son obligatorias, ya que se corresponden con funcionalidades secundarias de la aplicación y no se obliga al usuario a probarlas si no lo considera necesario.

  Para este bloque de la encuesta se han elaborado preguntas que se responden en base a una **escala de Likert**[1] del 0 al 10, donde 0 indica «nada de acuerdo» y 10 indica «muy de acuerdo» con la afirmación planteada. También se contempla en este bloque una pregunta (de respuesta obligatoria) sobre la satisfacción con el código generado, que se responde de forma cualitativa mediante una selección única entre varias opciones dadas.

- **Sugerencias y comentarios**: al final de la encuesta, se le proporciona al usuario la opción de dejar sugerencias o algún comentario si así lo desea. Esta parte de la evaluación es opcional y de respuesta libre. Si el usuario desea completarla, deberá plasmar sus comentarios o sugerencias en un cuadro de texto habilitado a tal efecto.

En el apéndice C de esta memoria pueden consultarse las preguntas concretas realizadas a los usuarios durante la evaluación.

8.3. Interpretación de los resultados

En esta sección se interpretan los resultados obtenidos de la evaluación con usuarios, desglosados en las categorías indicadas en el apartado anterior.

8.3.1. Datos personales

Los usuarios que han realizado la evaluación son todos mayores de edad. Se encuentran todos en una franja de edad entre los 18 y los 50 años, aunque más de la mitad de los resultados proceden de personas de entre 18 y 25 años de edad.

[1] Escala numérica que permite medir de forma cuantitativa cómo de acuerdo está la persona encuestada con una afirmación.

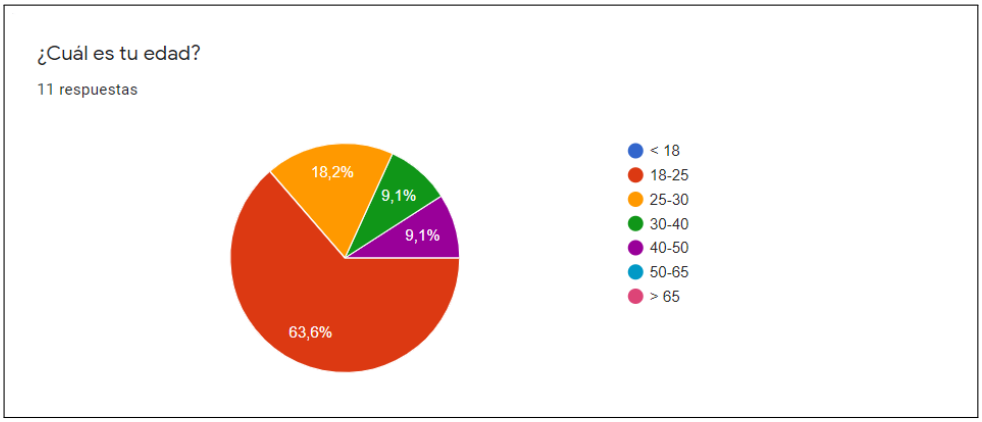

Figura 8.1: Distribución de la edad de los encuestados

En función de su género, un 72,7 % de los encuestados son hombres y el 27,3 % restante son mujeres, como puede apreciarse en el diagrama de sectores de la figura 8.2.

Acerca de su nivel de estudios, el 90,9 % de los encuestados afirma que

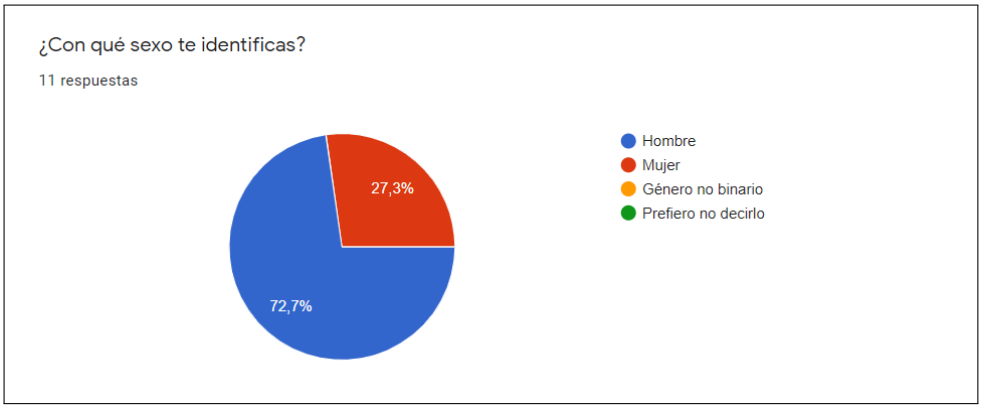

Figura 8.2: Distribución de los encuestados por género

cursa o ha cursado estudios universitarios, mientras que el 9,1 % de los votos restantes proviene de personas con estudios de Formación Profesional.

8.3.2. Conocimientos de bases de datos

Según los resultados obtenidos, aproximadamente 10 de los 11 encuestados han trabajado alguna vez con bases de datos. Véase la figura 8.4.

Como se representa en la figura 8.5, de estos usuarios que han trabajado

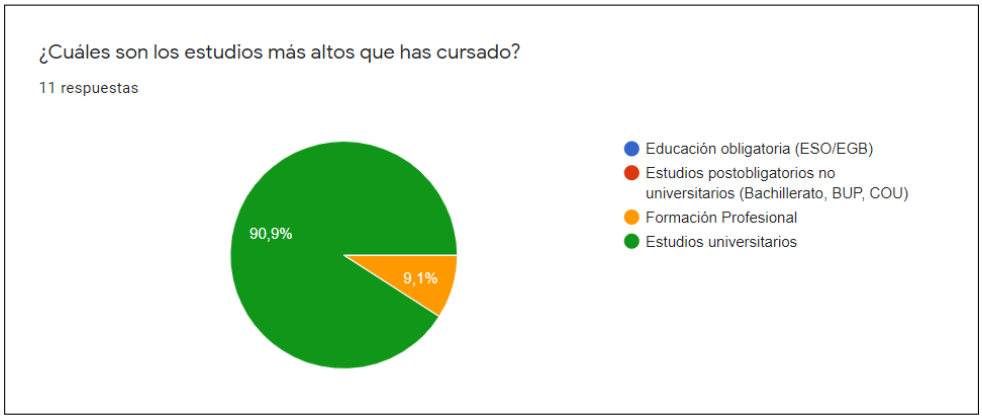

Figura 8.3: Distribución de los encuestados por nivel de estudios

al menos una vez con bases de datos, un 30 % solo está familiarizado con bases de datos SQL. En el otro extremo, el 70 % de los encuestados están habituados a trabajar con bases de datos de ambos tipos. Póngase especial atención en que no hay usuarios que hayan trabajado con bases de datos NoSQL exclusivamente.

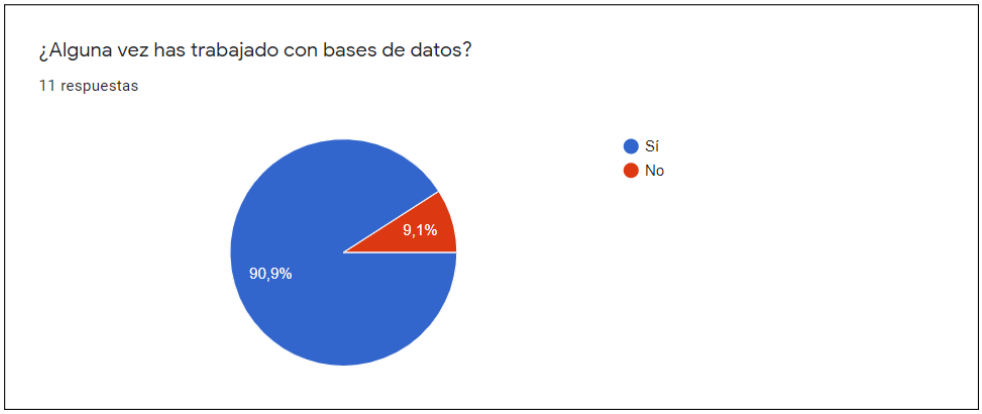

Figura 8.4: Distribución de los encuestados en función de si han trabajado con bases de datos o no

En la gráfica de la figura 8.6 se desglosan los usuarios habituales de bases de datos NoSQL en función de las plataformas sobre las que han trabajado alguna vez. Puede observarse que el lenguaje más popular es con diferencia MongoDB, seguido de Redis. Las bases de datos orientadas a columna y a grafo resultan ser las menos populares.

Con el objetivo de comprobar si la aplicación soluciona algún problema

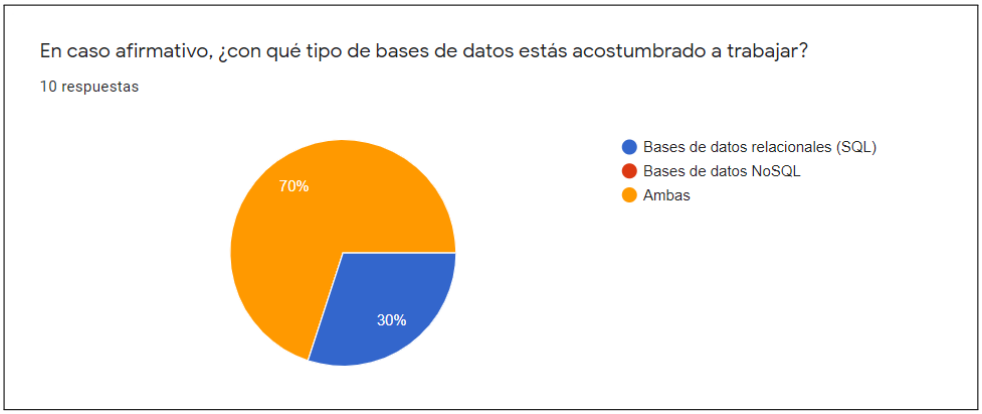

Figura 8.5: Distribución de los usuarios de bases de datos en función de la plataforma que usan habitualmente

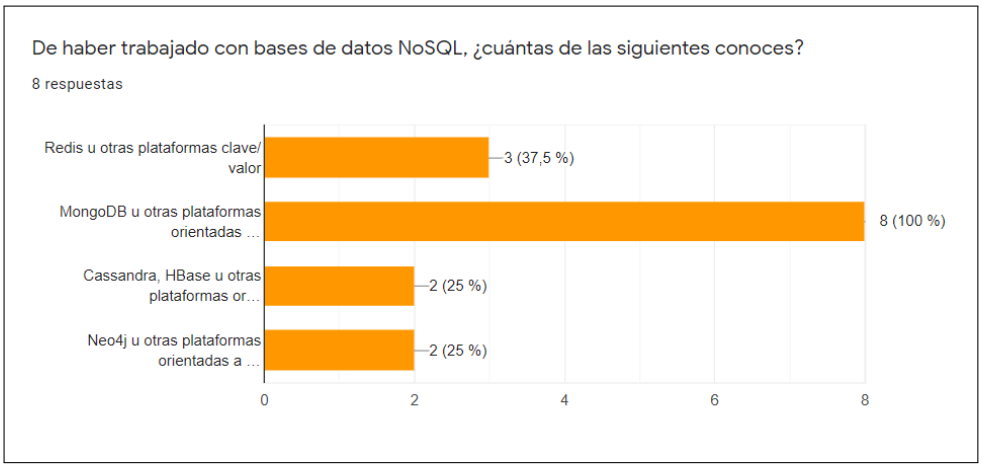

Figura 8.6: Plataformas NoSQL con las que han trabajado los encuestados

para los usuarios, se plantean una serie de problemáticas típicas de las bases de datos NoSQL, con el objetivo de que los miembros de la muestra examinada marquen aquellos con los que más se identifican.

Como puede comprobarse en la gráfica de la figura 8.7, el mayor problema que encuentran los usuarios es que apenas hay herramientas gráficas para el diseño.

8.3.3. Nivel de satisfacción con la aplicación

Tras recopilar información general sobre los usuarios evaluados, se procede a analizar los niveles de satisfacción de los mismos con el funcionamiento y el diseño del sistema.

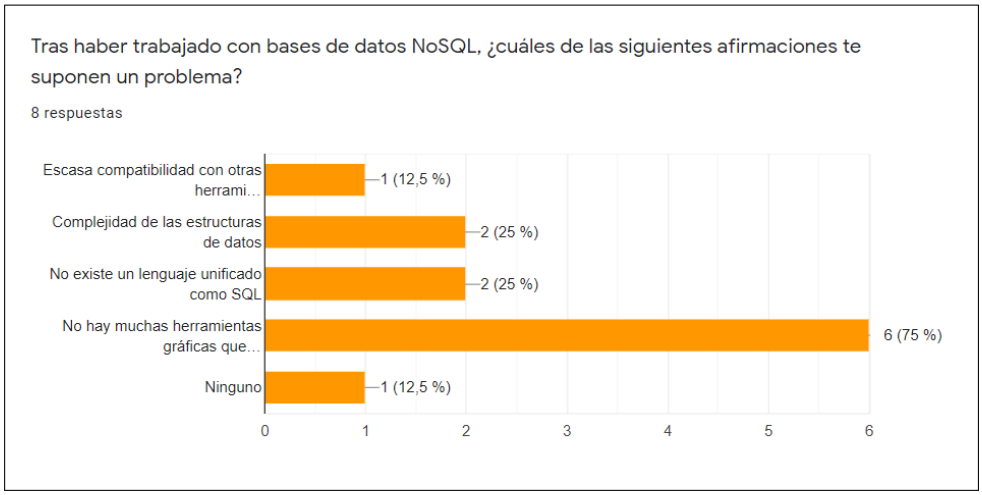

Figura 8.7: Distribución de los problemas típicos que encuentran los usuarios en las bases de datos NoSQL

En primer lugar se pregunta sobre el registro de usuarios, funcionalidad a la que todos los usuarios le dan una puntuación buena, entre 8 y 10 en todos los casos.

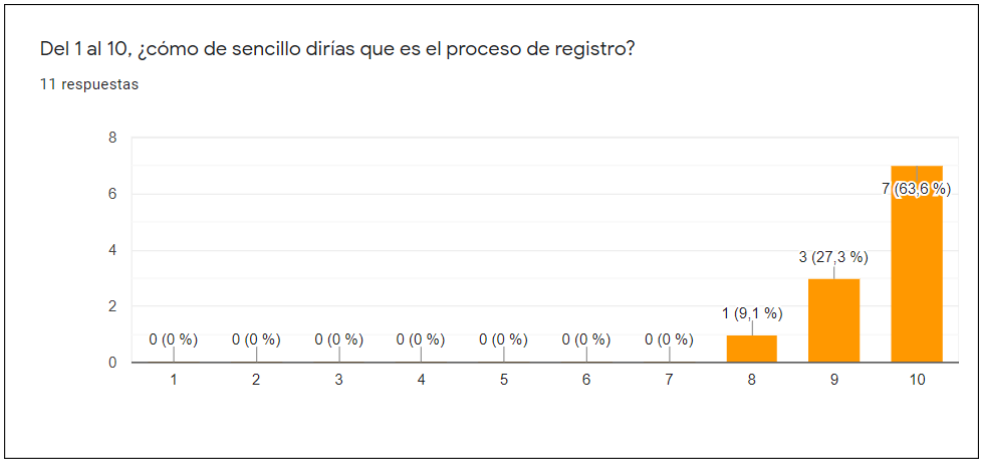

Figura 8.8: Distribución de los usuarios respecto a lo sencillo que les parece el procedimiento de registro

A la hora de iniciar sesión, todos los usuarios creen que es un procedimiento muy sencillo, como ilustran los resultados de la figura 8.9.

En el caso de que el usuario haya intentado recuperar su contraseña (este paso no es obligatorio para completar la evaluación), también se le pregunta

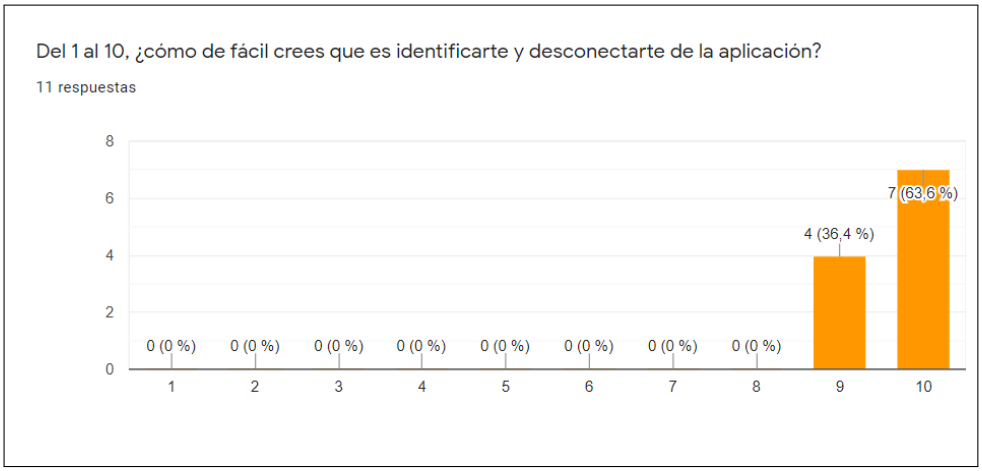

Figura 8.9: Distribución de los usuarios respecto a lo sencillo que les parece el procedimiento de login

sobre lo intuitivo que le ha parecido este paso. Del total de 11 usuarios encuestados, 8 de ellos han cambiado su contraseña, a los cuales les ha parecido bastante sencillo.

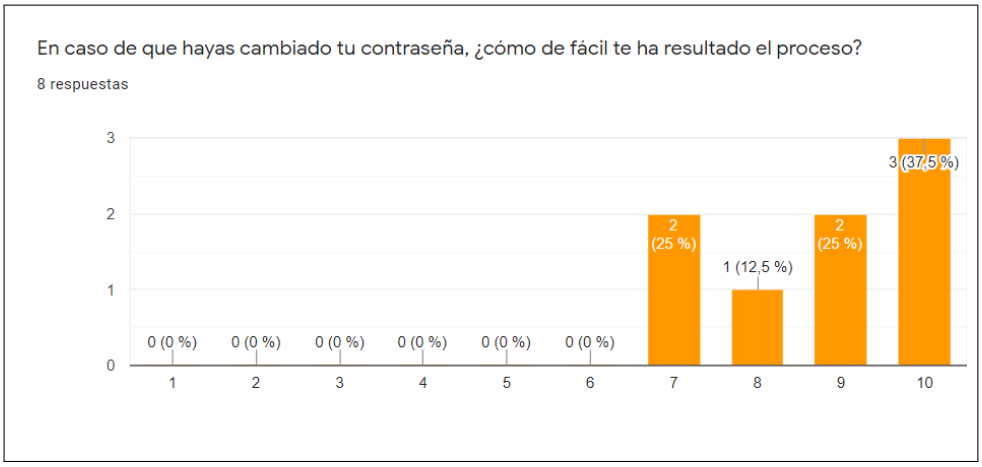

Figura 8.10: Distribución de los usuarios en función de lo intuitivo que creen que es cambiar la contraseña

En la figura 8.11 se exponen los resultados sobre lo sencillo que resulta consultar los datos personales desde la sección «Mi perfil». Como puede observarse, todos los usuarios coinciden en que la página es suficientemente intuitiva, si bien algunos opinan que la experiencia es mejorable.

Desde su apartado personal, el usuario podría haber escogido modificar

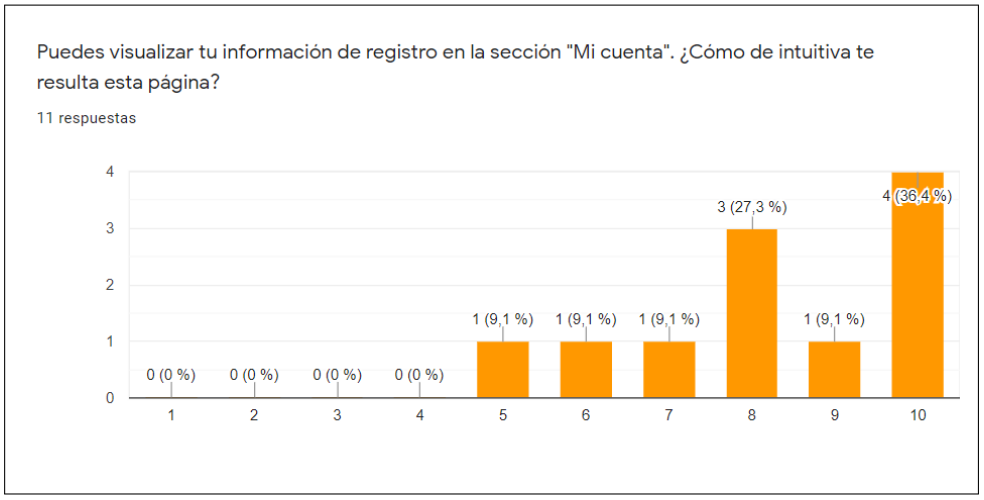

Figura 8.11: Distribución de los usuarios respecto a lo intuitiva que les resulta la página «Mi perfil»

sus datos de registro (tampoco es un paso obligatorio en la evaluación). Aun siendo un paso opcional, todos los usuarios encuestados han contestado a esta pregunta.

Los resultados obtenidos muestran que para casi todos los usuarios ha resultado sencillo, si bien hay un usuario que parece haber tenido algunas dificultades.

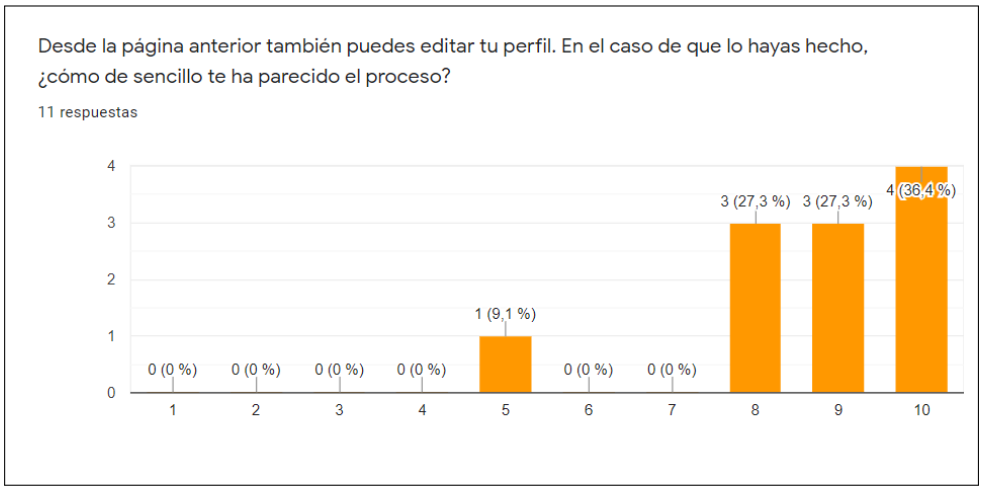

Figura 8.12: Distribución de los usuarios según las dificultades presentadas el editar su perfil

Respecto a la gestión de modelos desde el listado correspondiente, los

resultados apuntan a que es un proceso sin dificultades aparentes, ya que todos los encuestados le otorgan puntuaciones entre 7 y 10. Obsérvese la figura 8.13.

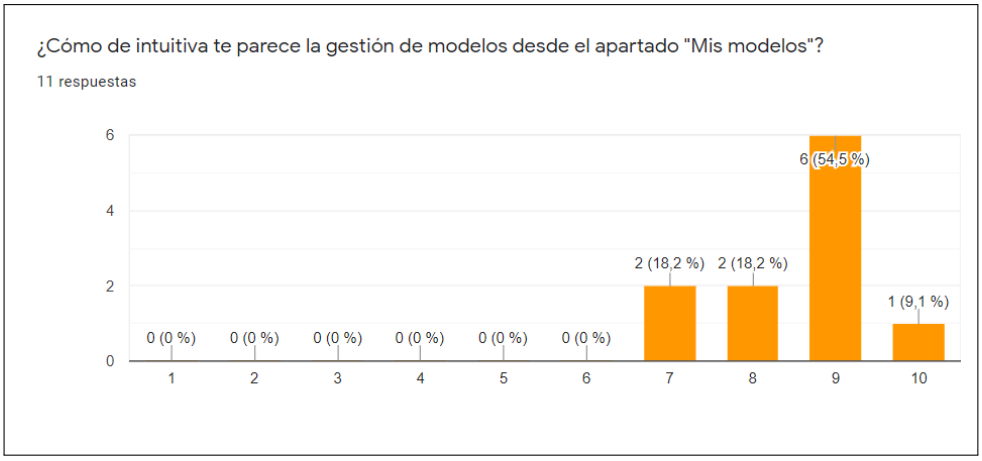

Figura 8.13: Distribución de los usuarios respecto a lo fácil que les resulta la gestión de modelos

Al ser preguntados sobre la facilidad del proceso de crear un nuevo modelo, más de la mitad de los usuarios evaluados han contestado que es un proceso muy sencillo. No obstante, algunos consideran que tiene algo de dificultad, aunque no en exceso.

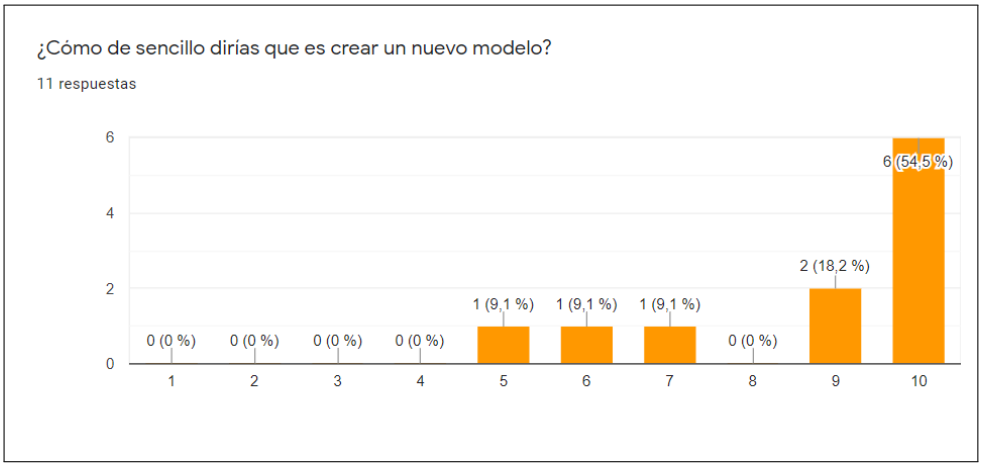

Figura 8.14: Distribución de los usuarios respecto a lo fácil que les resulta crear un nuevo modelo

Sobre lo intuitiva que es la interfaz gráfica del diseñador, un 18,2 % de los encuestados opinan que es insuficiente o mejorable. No obstante, el 81,8 % restante creen que, en mayor o menor medida, usar el diseñador es un proceso relativamente sencillo.

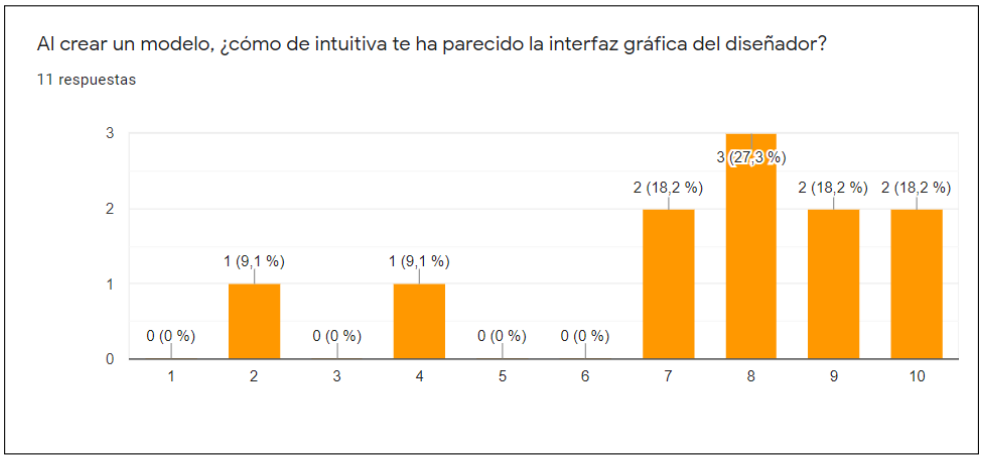

Figura 8.15: Distribución de los usuarios respecto a lo intuitivo que les resulta el diseñador

A la hora de utilizar las herramientas ofrecidas en el diseñador gráfico, 9 de cada 11 usuarios opinan que no supone un proceso complejo. Sin embargo, los 2 usuarios restantes han tenido alguna dificultad al operar sobre el lienzo de diseño, por lo que otorgan puntuaciones más bajas al diseño de las herramientas. Los datos detallados pueden consultarse en la figura 8.16.

Una vez que el usuario ya ha realizado algunas pruebas con modelos en el diseñador, se le invita a insertar información sobre el mismo. Sobre el procedimiento a seguir, más de un 80 % de los usuarios afirman que es un procedimiento sencillo. Aun así, llama la atención que para un usuario ha sido excesivamente difícil completar este paso.

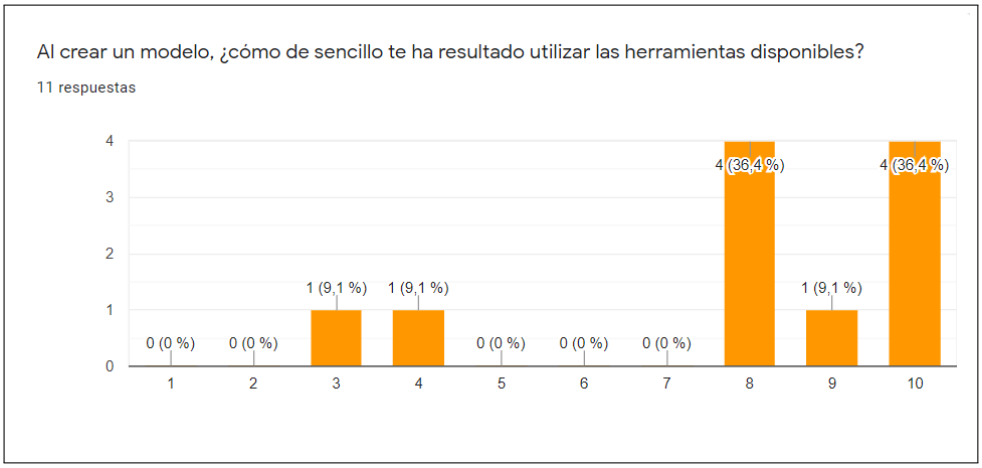

Figura 8.16: Distribución de los usuarios respecto a lo fácil que les resulta usar las herramientas del diseñador

Dentro de la misma página del formulario de inserción de datos, los usuarios están de acuerdo en que es fácil o muy fácil modificar información en documentos ya existentes o eliminar documentos registrados. Así lo ilustran los resultados, expuestos en la figura 8.18.

Tras insertar algunos documentos, se pide a los usuarios evaluados que valoren la página para guardar el modelo y exportar los resultados (la página que genera el código para MongoDB). En general, los resultados obtenidos son buenos. Todos los usuarios aprueban el funcionamiento de esta página, y más de la mitad opinan que es muy intuitiva. Véase la figura 8.19.

Para encontrar posibles errores en la implementación de la aplicación además de en el diseño, se solicita a los usuarios que indiquen si el código generado era el esperado. Todos los usuarios que están familiarizados con la sintaxis de MongoDB afirman que el código generado fue correcto, mientras que otro porcentaje más pequeño no está seguro porque no conoce el lenguaje. Obsérvense los resultados detallados en la figura 8.20.

Como se muestra en la figura 8.21, el nivel general de satisfacción de los usuarios con el diseño de la aplicación es bueno. Solo un 9,1 % de los usuarios evaluados consideran que es insuficiente, mientras en el otro extremo, un 45,5 % de usuarios afirman que están muy satisfechos con el diseño.

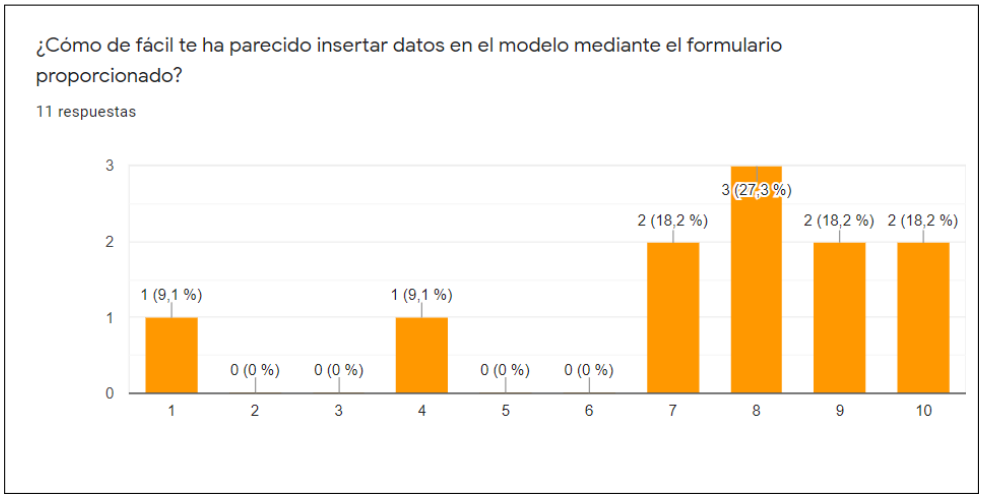

Figura 8.17: Distribución de los usuarios respecto a la dificultad expresada para insertar datos en un modelo

A niveles generales, todos los usuarios aprueban el funcionamiento de la aplicación. Inclusos algunos de ellos se consideran bastante o muy satisfechos, ya que el 90,9 % le otorga puntuaciones de 7 a 10. Véanse los resultados en la figura 8.22.

8.3.4. Sugerencias y comentarios

Como se comenta en la sección que describe la estructura de la encuesta, al final de la misma se deja la opción de que los usuarios dejen comentarios o aporten sugerencias. Del total de 11 usuarios evaluados, solo 3 de ellos han querido hacer puntualizaciones:

Como puede apreciarse en los comentarios de la figura 8.23, se proponen mejoras concretas para la página del diseñador gráfico. En particular se sugiere mejorar la escalabilidad del zoom y facilitar el redimensionado de los nodos.

Además, un usuario puntualiza que ha realizado la evaluación desde un dispositivo móvil y que puede que haya influido negativamente en sus resultados.

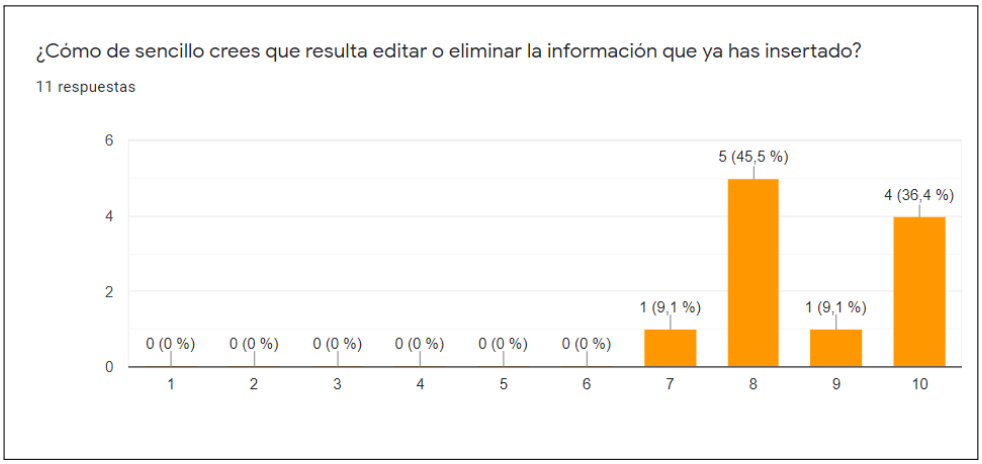

Figura 8.18: Distribución de los usuarios respecto a la dificultad expresada para modificar datos insertados en un modelo

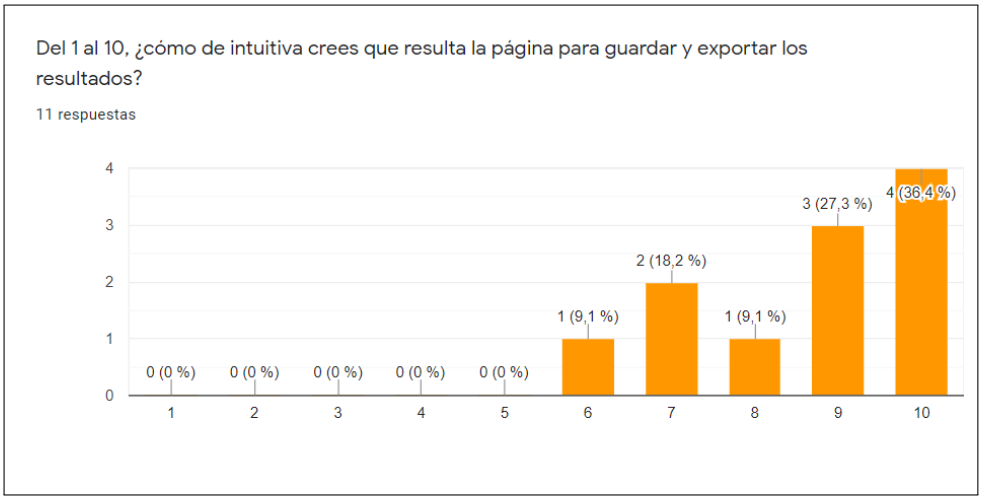

Figura 8.19: Distribución de los usuarios según lo intuitiva que consideran la página «Guardar y exportar»

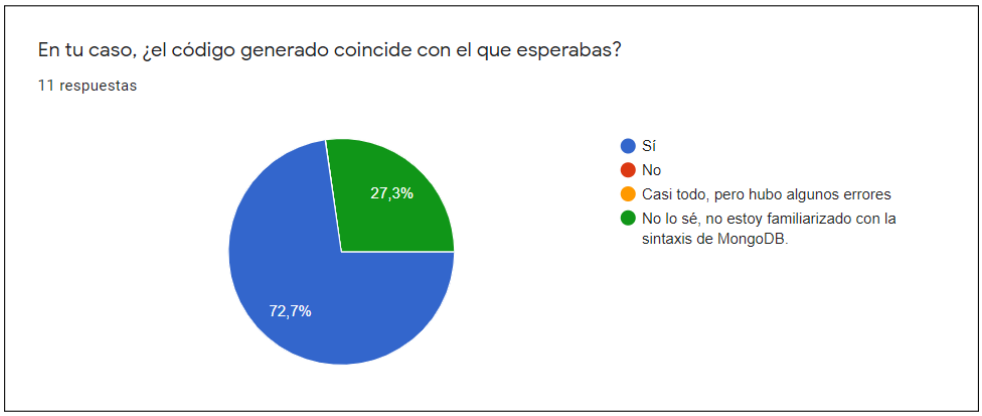

Figura 8.20: Satisfacción de los usuarios con el código generado

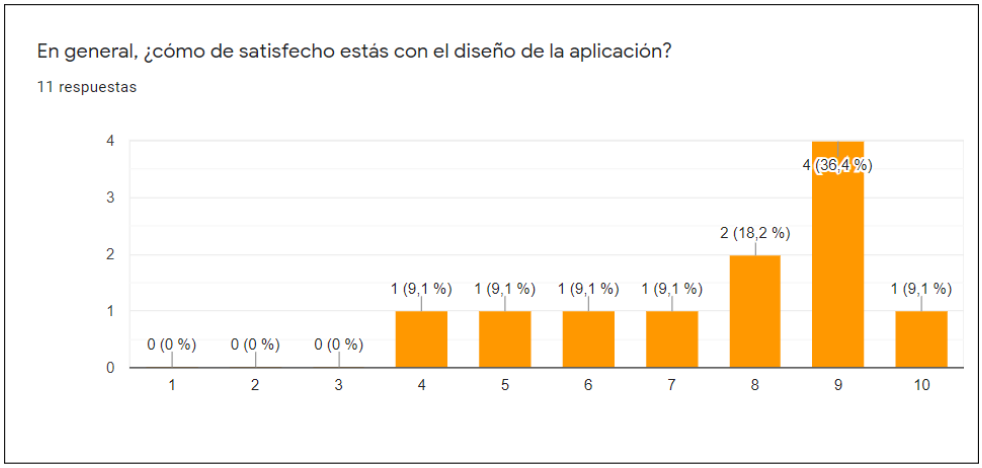

Figura 8.21: Satisfacción general de los usuarios con el diseño de la aplicación

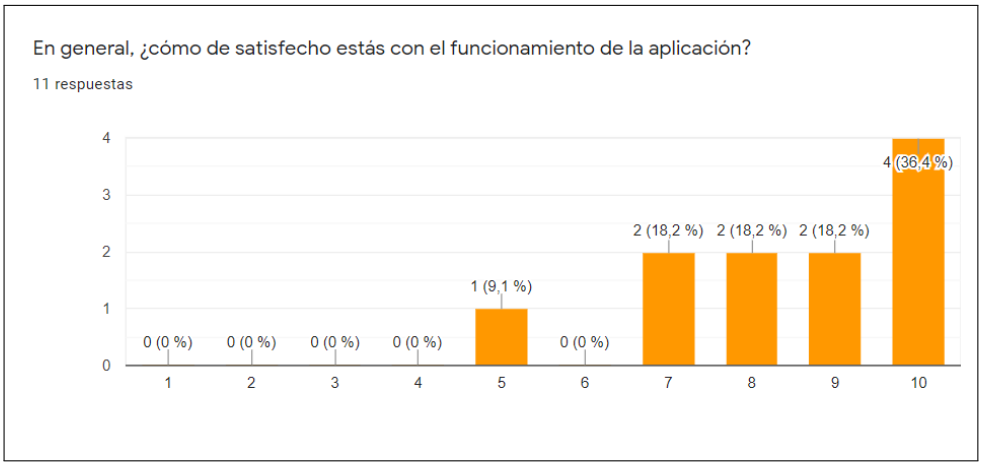

Figura 8.22: Satisfacción general de los usuarios con el funcionamiento de la aplicación

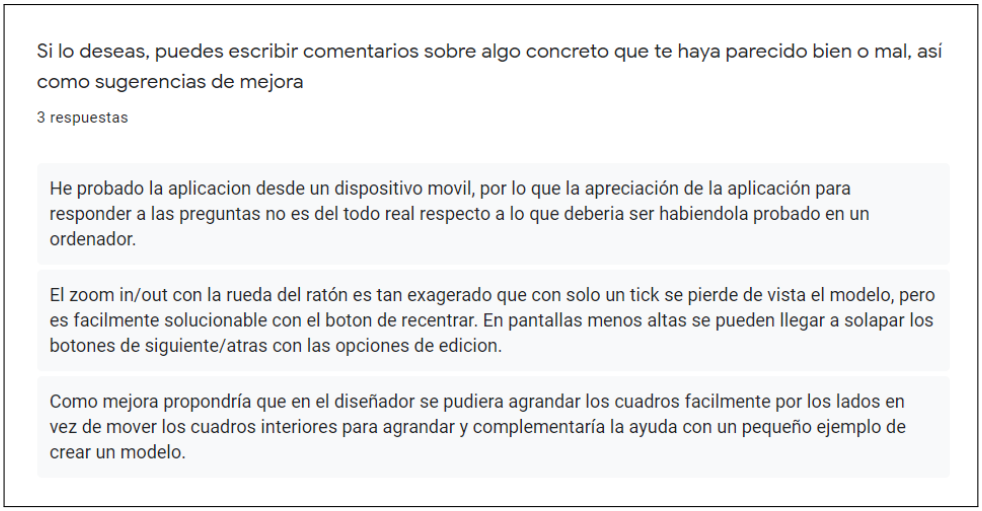

Figura 8.23: Comentarios y sugerencias de los usuarios encuestados

Capítulo 9

Conclusiones y trabajo futuro

No puedes volver atrás y cambiar el principio, pero puedes comenzar donde estás y cambiar el final.

C.S. Lewis

9.1. Introducción

En este capítulo se exponen las conclusiones deducidas tras el desarrollo del presente Trabajo de Fin de Grado. De igual manera, se exponen también las posibles líneas de trabajo futuro en este proyecto.

9.2. Conclusiones del trabajo

En este Trabajo de Fin de Grado se ha desarrollado una aplicación web que permite modelar bases de datos NoSQL de una forma gráfica. Con la implementación propuesta se han cumplido todos los objetivos planteados de forma previa al desarrollo, expuestos en el capítulo 1 de esta memoria. En concreto:

- A través del diseñador gráfico se ha conseguido que los usuarios puedan generar modelos NoSQL sin tener conocimientos específicos.
- Los modelos generados son almacenados en la base de datos de la aplicación para asegurar la persistencia de los mismos, evitando tener que descargar los resultados y tener que importarlos para realizar cambios.
- Mediante el generador de código para MongoDB, se ha logrado exportar los resultados del modelo creado a una plataforma concreta.

9.3. Trabajo futuro

Si bien se han implementado todos los casos de uso que se recogen en la especificación de requisitos, sería interesante plantear como posibles líneas de trabajo futuro las que se listan a continuación:

- **Mejorar la interfaz gráfica del diseñador**. Tras una evaluación con usuarios, se ha demostrado que los usuarios son capaces de utilizar todas las funciones implementadas, aunque en algunos casos ponen de manifiesto que su uso es poco intuitivo.
- **Generar código para otras plataformas NoSQL** que estructuran su modelo en base a agregados de datos. Por ejemplo, para Cassandra, una de las principales plataformas NoSQL orientadas a columna.
- **Añadir un diseñador de consultas**. El nivel de implementación actual permite que el usuario diseñe una base de datos en MongoDB sin conocer su sintaxis. Sin embargo, si no está familiarizado con la sintaxis, le resultará complicado ejecutar consultas sobre la base de datos creada. Por tanto, sería una idea interesante permitir al usuario diseñar las consultas de una forma también gráfica. Por ejemplo, aplicando filtros con casillas de verificación.
- **Mejorar la eficiencia de las consultas** a la base de datos. Actualmente, el modelo de datos es muy simple: no tiene índices ni otros mecanismos que aceleren la ejecución de las consultas. Sin embargo, de cara a facilitar la escalabilidad de la aplicación, se deberían optimizar las consultas.
- **Aumentar las funciones de los administradores** del sistema. Hasta el momento, solo pueden realizar operaciones relativas a la administración de usuarios, pero no gestionar los modelos de dichos usuarios.
- **Mejoras generales** y detección y corrección de errores. Es crucial realizar una labor de mantenimiento del software para que la experiencia de usuario no se vea desmejorada con el paso del tiempo.

Chapter 9

Conclusions and future work

You can't go back and change the beginning, but you can start where you are and change the ending.

C.S. Lewis

9.1. Introduction

The conclusions gotten from the development of this Final Project are explained in this chapter. In addition, future lines of work are proposed for the project.

9.2. Conclusions of the project

A web application has been developed for this project. This application allows to model NoSQL databases in a graphic way. The implementation described has made possible that all the objectives purposed in chapter 1 were achieved. In particular:

- The users are able to generate NoSQL models without having specific knowledges, which can be carried out through the graphic designer.
- The generated models are stored in the application database to ensure the persistence of the data. This avoids the user has to download their results and loading them to the application to make changes.
- The results can be exported to a specific platform. This work is done by the MongoDB code generator.

9.3. Future work

Even though all the use cases listed in the software requirements specification have been implemented, it would be interesting to bring up as possible future lines of work on this project the following ones:

- **Improving the graphic user interface of the designer**. After doing an evaluation on user experience, the users have demonstrated that they are able to use all the application functionalities. However, the poll reveals that, in some cases, these functionalities were not as intuitive as expected.
- **Generating code for other NoSQL platforms** that use aggregate-based models. For example, Cassandra, one of the most popular column-oriented systems.
- **Incorporating a query designer**. The current level of implementation allows the user to design a MongoDB database without knowing the syntax. However, if the users are not familiar to the syntax, they will found difficulties to build the statements they need to query the database. Because of this, it would be a good idea to provide the user a way to design these statements in a graphic way as well. For example, this could be done by applying filters using checkboxes.
- **Improving the efficency of the queries to the database**. Currently, the application data model is very simple: it does not have indexes or any mechanisms that speed up the execution of the queries. Nevertheless, the queries should be optimized in order to ease the scalability of the application.
- **Increasing the functions of the system administrators**. At the moment of writing this report, administrators can manage user accounts only, and not the models created by the users.
- **General improvements and bugs fixing**. It is crucial to mantain the software so the pass of time does not degrade the user experience.

Apéndice A

Guía de instalación de la aplicación

A.1. Introducción

Este apéndice conforma una guía paso a paso de las acciones necesarias para instalar el código de la aplicación desarrollada en un servidor web, así como el procedimiento necesario para configurar la base de datos de la aplicación.

Para la elaboración de esta guía, la aplicación ha sido instalada en el sistema operativo **Debian 10**, aunque la instalación en otras distribuciones Linux es similar. No obstante, se recomienda utilizar dicha distribución en un entorno virtualizado.

De realizar la instalación en una máquina virtual, es responsabilidad del lector asegurarse de que la misma tiene conexión a internet; en otro caso no podrá ejecutar la funcionalidad de recuperar las contraseñas de usuario.

Todo el código necesario para instalar la aplicación está disponible de forma pública en un repositorio GitHub[1] mantenido por el autor.

A.2. Instalar y configurar el servidor

En primer lugar se instalará el servidor web de Apache. Simplemente hay que abrir una terminal y teclear los siguientes comandos:

```
$ sudo apt update && sudo apt -y upgrade
```

[1] `https://github.com/rmdez/tfg`

```
$ sudo apt install apache2
```

También es necesario configurar un servidor de correo para enviar los emails de recuperación de contraseña. En este caso la aplicación se instala en local, por lo que será necesario enviar el correo a través de un servidor externo que sí es «real». Para ello se instalará un paquete y se abrirá el puerto 465 (SMTP con SSL) para poder conectarse al servidor de correo:

```
$ sudo apt-get install sendmail
```

```
$ sudo iptables -A OUTPUT -p tcp --dport 465 -j ACCEPT
```

Una vez configurado el servidor, se procede a la instalación del intérprete de código PHP.

A.3. Instalar intérprete PHP

El primer paso en la instalación de PHP será insatalar el paquete general del intérprete:

```
$ sudo apt -y install php php-common
```

A continuación se instalan las librerías y extensiones de PHP que se necesitan para trabajar con la aplicación[2]:

```
$ sudo apt -y install php-cli php-fpm php-json php-pdo php-mongodb
php-zip php-gd php-mbstring php-curl php-xml php-pear php-bcmath
```

```
$ sudo apt -y install libapache2-mod-php
```

```
$ sudo a2enmod php7.3
```

Si esta última instrucción no se ejecutase correctamente, comprobar primero la versión de PHP instalada (puede diferir de la 7.3) con el siguiente comando:

```
$ sudo php -v
```

[2]Se recomienda copiar los comandos de varias lineas en un editor de texto antes de introducirlos en la consola, ya que se pueden copiar los saltos de línea y ejecutar solo la primera línea del comando, dando lugar a errores.

Si la versión de PHP que se ha instalado fuera distinta de la 7.3, simplemente hay que cambiar el número de versión en la instrucción anterior. Procedemos a reiniciar el servicio de Apache para que los cambios surtan efecto:

```
$ sudo systemctl restart apache2
```

A.4. Instalar MongoDB Community Server

A continuación es necesario instalar el software de servidor de MongoDB. Nótese que la instalación descrita en esta sección es **específica para Debian 10**. En el caso de estar utilizando otra distribución Linux, consultar las instrucciones correspondientes en la documentación oficial de MongoDB.[3].

Para realizar el proceso de instalación, ejecútense los siguientes comandos en una terminal:

```
$ wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc
| sudo apt-key add -
```

En el caso de que `gnupg` no esté instalado en el sistema, la ejecución del comando anterior resultará en un estado de error. Para solucionarlo, primero hay que instalar `gnupg` con el comando que se indica a continuación, y volver a ejecutar la orden anterior.

```
$ sudo apt-get install gnupg
```

Prosiguiendo con la instalación de MongoDB:

```
$ echo "deb http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2
main"| sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install -y mongodb-org
```

```
$ sudo systemctl start mongod
```

La última instrucción de este bloque inicia MongoDB como servicio. Sin embargo, como se acaba de instalar, es posible que el sistema aún no encuentre el servicio. Para solucionar este problema, lo único que hay que hacer es

[3] `https://docs.mongodb.com/manual/administration/install-on-linux/`

refrescar la lista de demonios del sistema y volver a lanzar el servicio:

```
$ sudo systemctl daemon-reload

$ sudo systemctl start mongod
```

A.5. Copiar el código fuente al directorio del servidor

Para terminar de instalar la aplicación, tan solo hay que copiar los archivos de la misma al directorio raíz del servidor. En este directorio hay un fichero HTML de ejemplo, que borraremos para no integrarlo con el resto de la aplicación. Para ello se ejecutará la siguiente orden:

```
$ sudo rm -r /var/www/html/*
```

Podemos comprobar a continuación que el directorio ha quedado, en efecto, vacío:

```
$ sudo ls -la /var/www/html
```

A continuación, solo queda copiar los archivos a ese mismo directorio:

```
$ sudo cp -r path_directorio_origen/* /var/www/html/
```

En ocasiones, al copiar ficheros al directorio `/var/www/` se copian sin permisos de lectura o ejecución y el servidor podría tener problemas para cargar la aplicación. Para prevenir esto, es recomendable modificar los permisos del directorio con el siguiente comando:

```
$ sudo chmod -R 0755 /var/www/
```

A.6. Importar información en la base de datos

El último paso es cargar la información en la base de datos. Para ello, obtener el volcado de la base de datos del repositorio del proyecto y ejecutar el siguiente comando:

```
$ mongorestore -d nosqldd path_repositorio/includes/
databaseDump/nosqldd
```

La aplicación ya está instalada y es accesible desde *localhost*. Para empezar a utilizarla, simplemente se abrirá el navegador web deseado y se escribirá `localhost` o su IP homónima `127.0.0.1` en la barra de direcciones.

Apéndice B

Guía de uso de la aplicación

B.1. Introducción

Para complementar el contenido de esta memoria, en este apéndice se presenta una guía de uso para utilizar la aplicación desarrollada. Mediante un caso práctico se explica cómo interactuar con cada funcionalidad, detallando las entradas que espera el sistema del usuario y cuáles son las salidas esperadas.

B.2. Página de inicio

La página de inicio (`index.html`) es la primera que se visualiza al llegar al sitio web. Aquí se puede encontrar una galería de imágenes que resumen las características principales de la aplicación. Puede navegarse por las imágenes haciendo click sobre los controles que aparecen a la izquierda y derecha de las mismas.

Para empezar a utilizar el resto de funcionalidades de la aplicación, es necesario estar identificado como usuario. Por tanto, si no se tiene una cuenta, es necesario hacer click en el botón «Registrarse» para crear una.

B.3. Registro de usuarios e inicio de sesión

En esta sección se muestra el funcionamiento de todos los procesos relativos a las cuentas de usuario.

B.3.1. Registro de usuario

Una vez alcanzada la página `register.php`, se mostrará en pantalla un formulario que se debe rellenar para crear una cuenta. Tras introducir los

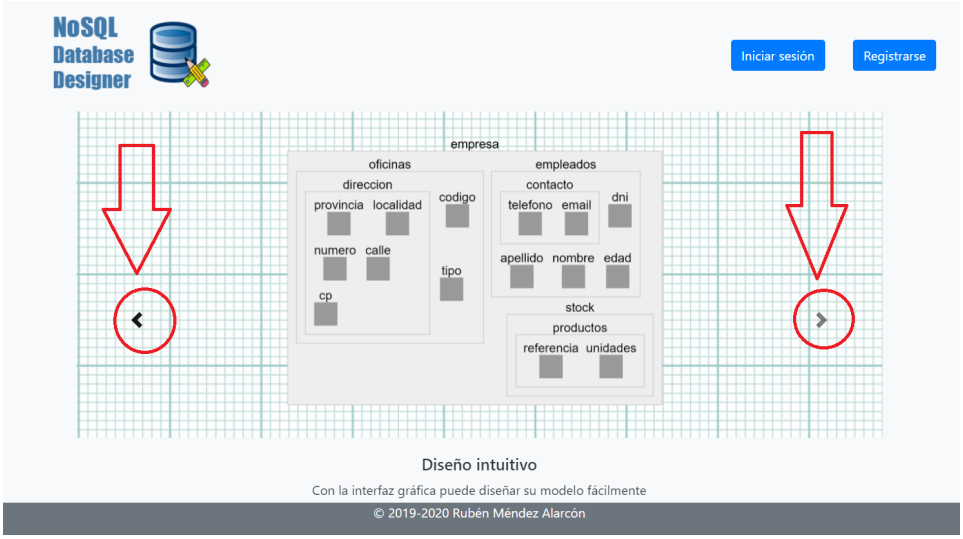

Figura B.1: Ubicación de los controles del carrousel en la página de inicio

datos personales, solo hay que pulsar el botón enviar para finalizar el proceso. En el caso de que los datos introducidos fueran incorrectos (por ejemplo, que las contraseñas no coincidan) se mostrará un error y será necesario repetir la operación. Por otro lado, si los datos introducidos son correctos, se creará la cuenta de usuario y se iniciará la sesión automáticamente.

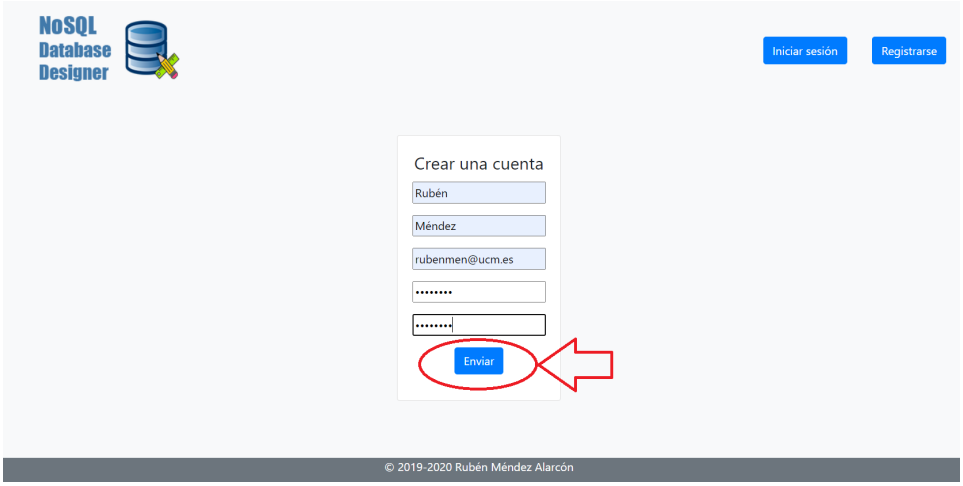

Figura B.2: Formulario de registro cumplimentado con datos de ejemplo

B.3.2. Login o inicio de sesión

Si por el contrario ya se había creado una cuenta anteriormente, se hará click en el botón «Iniciar sesión» de la barra de navegación, que redirige al usuario a la página `login.php` para introducir sus credenciales.

Se muestra un formulario que requiere una dirección de correo electrónico válida y la contraseña de dicho usuario. Tras introducir los datos es necesario enviar el formulario para procesar los datos. Si son correctos, la sesión quedará iniciada. En otro caso, se notifica que los datos son incorrectos y se muestra la opción de recuperar la contraseña.

Figura B.3: Formulario de login cumplimentado con datos de ejemplo

B.3.3. Recuperación de contraseña

Tras realizar un intento de login incorrecto, se muestra la posibilidad de recuperar la contraseña. Para ello, hay que hacer click en el enlace proporcionado.

En esta página, el usuario encontrará un formulario con un único campo, donde tiene que introducir la dirección de correo con la que se registró. Si la dirección es correcta, se enviará un email con un enlace único para cambiar la contraseña.

Al hacer click en el enlace, el usuario será redirigido a la página `changePwd.php` de la aplicación, donde debe confirmar de nuevo su correo electrónico e introducir la nueva contraseña. Si al enviar el formulario los datos son correctos, se cambiará la contraseña antigua por la indicada. En caso contrario, se muestra el error.

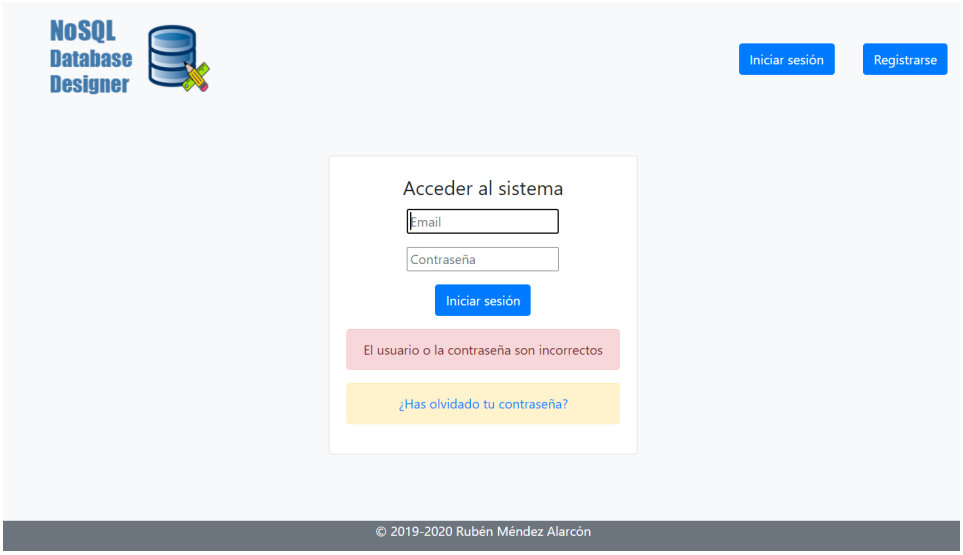

Figura B.4: Resultado de un intento incorrecto de login

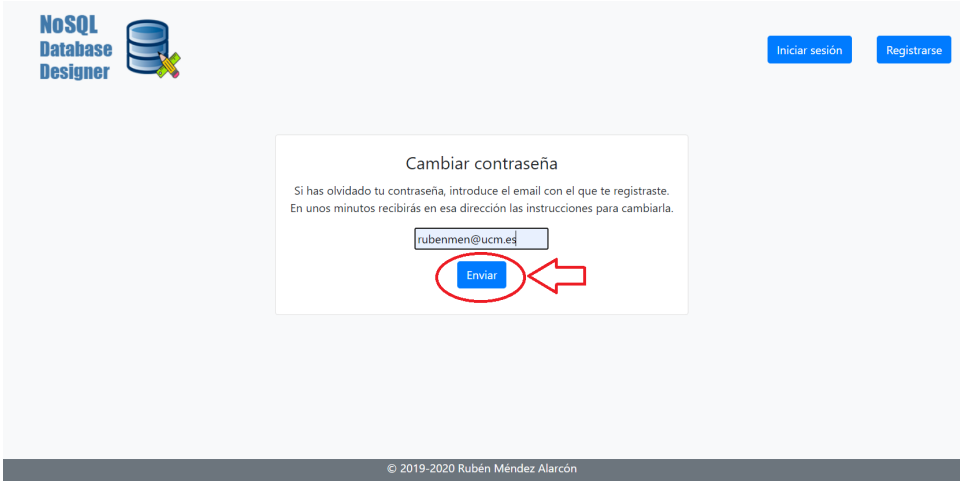

Figura B.5: Formulario de solicitud de nueva contraseña cumplimentado con datos de ejemplo

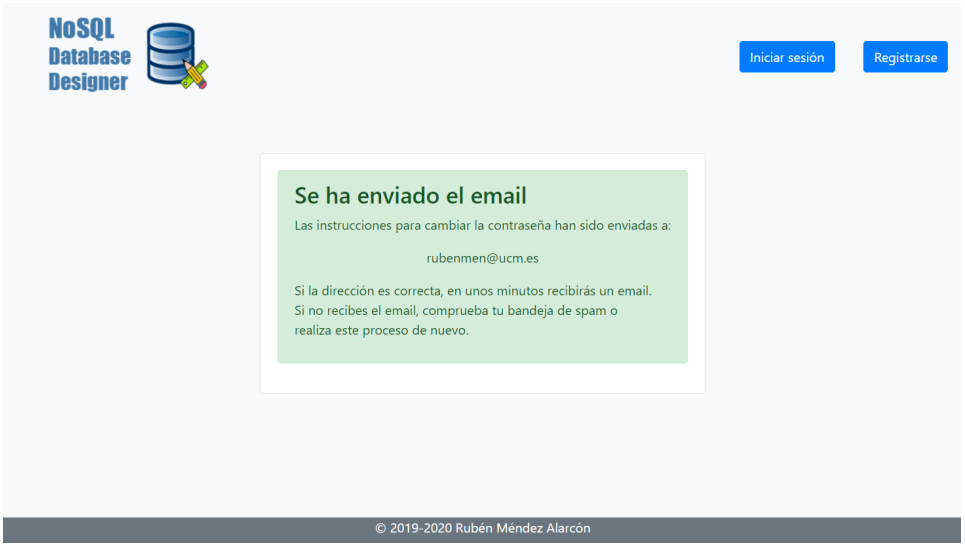

Figura B.6: Mensaje de confirmación tras enviarse el email para cambiar la contraseña

Figura B.7: Contenido del email recibido para cambiar la contraseña

Figura B.8: Formulario de cambio de contraseña cumplimentado con datos de ejemplo

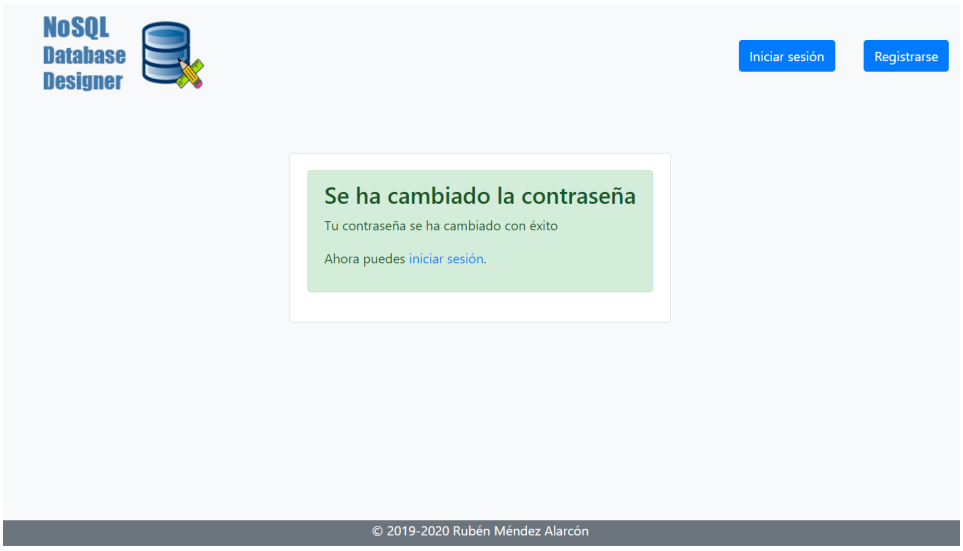

Figura B.9: Mensaje de confirmación tras cambiar la contraseña

B.4. Creación y gestión de proyectos

Para ilustrar el proceso de creación de un proyecto, así como las operaciones que se pueden realizar sobre los mismos, en esta sección se muestra paso a paso la creación de algunos modelos de ejemplo.

B.4.1. Crear proyecto

Una vez iniciada la sesión, el usuario es redirigido a su listado de modelos (página `databases.php`). De no ser así, se puede acceder fácilmente a esta página haciendo click en el botón «Mis modelos» de la barra superior de navegación.

A continuación se abrirá un cuadro de diálogo en el que se debe introducir el nombre del modelo, que no puede contener espacios ni caracteres especiales. Si el nombre es válido, se creará el modelo y se cargará el lienzo de diseño (cuyo uso se detalla más tarde en este apéndice). En otro caso, se muestra otro cuadro de diálogo con el error y es necesario repetir el proceso.

Si se hace click en «Mis modelos» para volver al listado de diseños, este ya no aparecerá vacío, sino que mostrará que hay un modelo con cambios sin guardar.

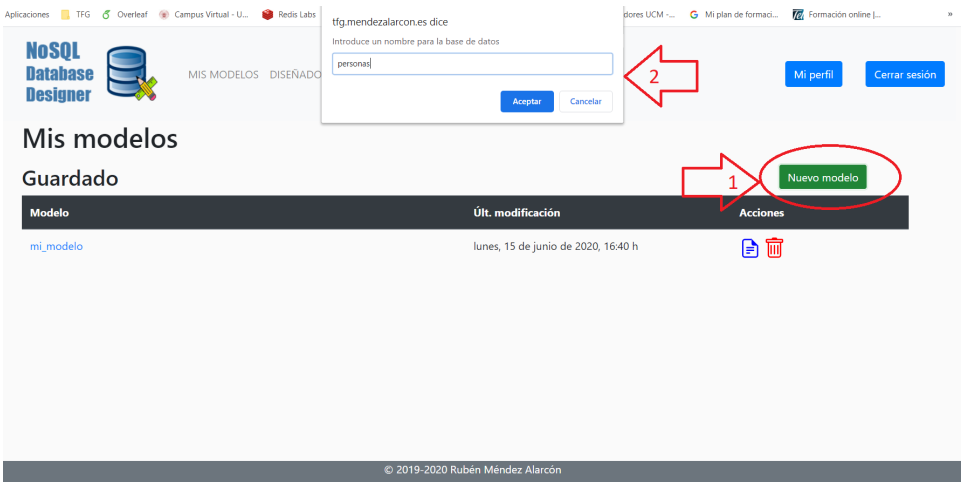

Figura B.10: Pasos a seguir para crear un nuevo proyecto o modelo

B.4.2. Descartar cambios

Cuando un modelo está abierto, se puede cerrar guardando los cambios o descartándolos. Si se elige por descartar los cambios, se perderán todas las modificaciones llevadas a cabo desde que se abrió el modelo.

Para descartar los cambios realizados sobre un modelo, tan solo es necesario hacer click en el botón correspondiente y confirmar la operación.

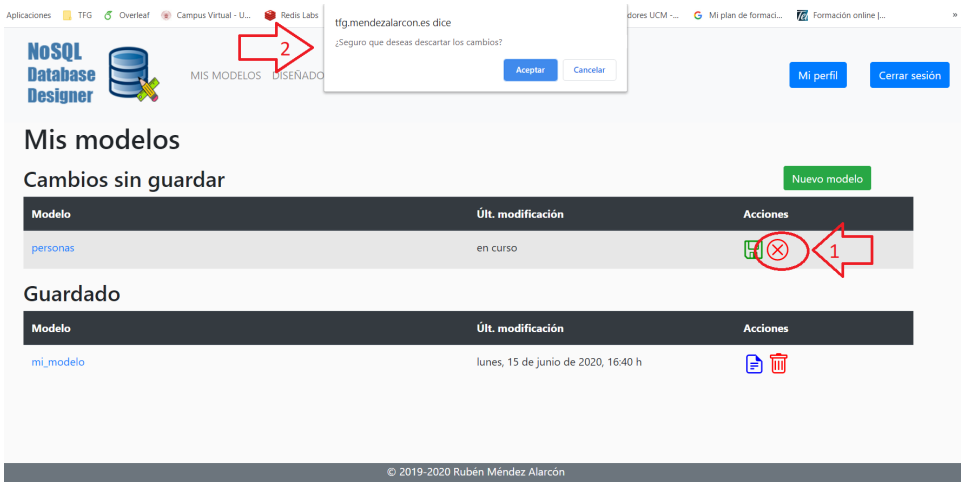

Figura B.11: Pasos a seguir para descartar los cambios sobre un borrador

B.4.3. Guardar cambios

La otra opción para cerrar un modelo en edición es guardar los cambios realizados. Para hacer efectivos los cambios, solo es necesario hacer click en el botón «Guardar cambios» (representado con un icono de disquete verde). El modelo aparecerá entonces en el listado de proyectos guardados.

B.4.4. Cambiar nombre

Permite cambiar el nombre del modelo después de su creación. Para cambiar el nombre de un modelo, solo hay que hacer click en el botón «Cambiar nombre» del modelo deseado e introducir un nuevo nombre válido. Las restricciones de nombre son las mismas que al crear un nuevo modelo.

B.4.5. Eliminar modelo

Si no se desea volver a trabajar con un modelo, puede eliminarse del sistema haciendo uso del botón eliminar. No hay más que pulsar el botón «eliminar» del modelo deseado y confirmar la operación. El modelo desaparecerá del listado. Esta acción es **irreversible**.

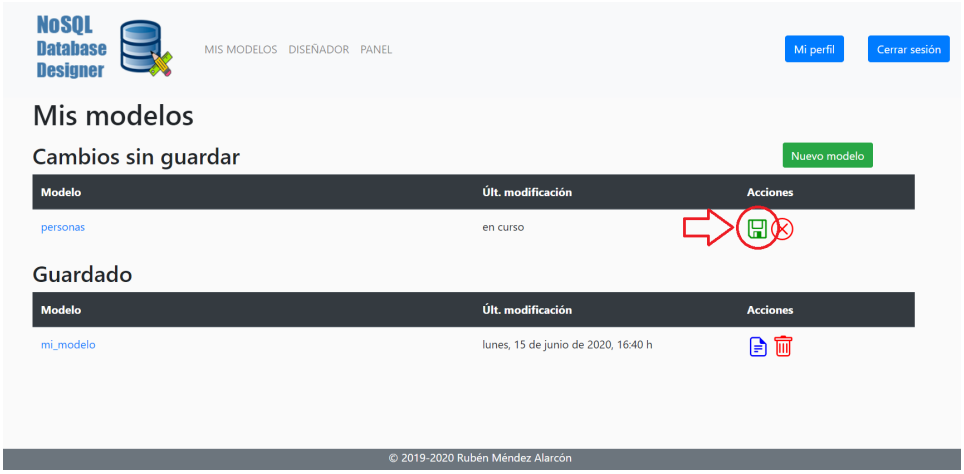

Figura B.12: Pasos a seguir para guardar un modelo en edición

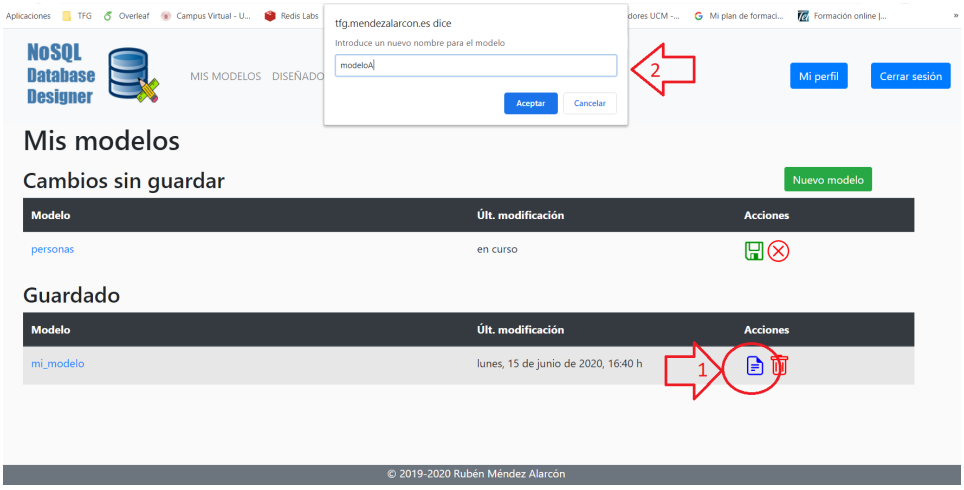

Figura B.13: Pasos a seguir para cambiar el nombre de un modelo

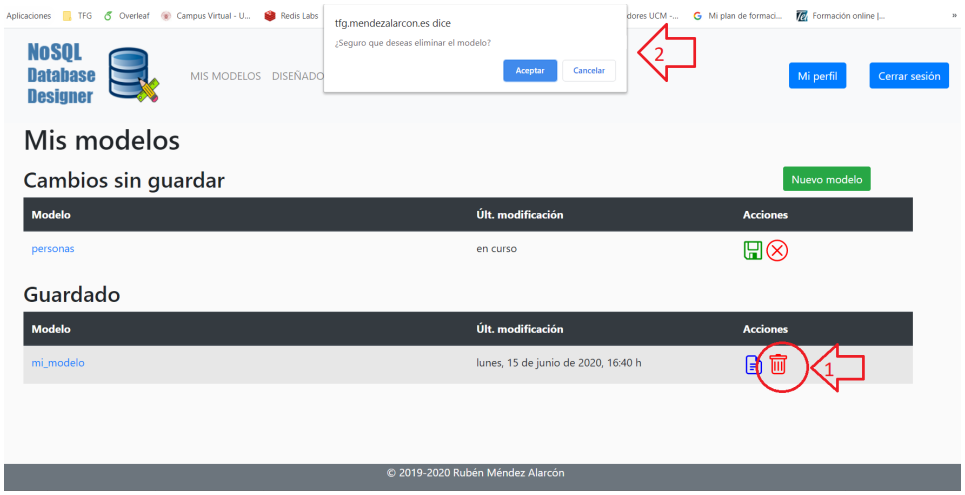

Figura B.14: Pasos a seguir para eliminar un modelo guardado

B.5. Creación de modelos

Una vez inicializado un proyecto, se puede comenzar a diseñar el modelo desde la página «Diseñador». Al crear un proyecto, por defecto se añade un item al lienzo, dentro del cual se debe realizar el resto del modelo.

B.5.1. Añadir item

Para crear un modelo en base a agregados, es necesario anidar items dentro de otros. Por lo tanto, para agregar items de primer nivel, hay que seleccionar el item que representa el modelo y hacer click en el botón correspondiente. Se abrirá un cuadro de diálogo para introducir el nombre de ese item.

Si se está añadiendo un item de primer nivel, no podrá contener espacios ni caracteres especiales (por las restricciones que hay al exportar el modelo a la sintaxis de MongoDB). Los items de primer nivel deberán de ser agregados que contegan otros items (lo que dará lugar a colecciones en MongoDB).

Dentro de esos agregados, pueden añadirse tanto items individuales (que serán campos rellenables) como otros agregados. El proceso es el mismo: seleccionar el item contenedor y pulsar el botón «añadir item».

B.5.2. Eliminar item

Para eliminar un item, simplemente hay que seleccionarlo haciendo click y pulsar el botón eliminar. Si ese item ya tenía información asociada, habrá

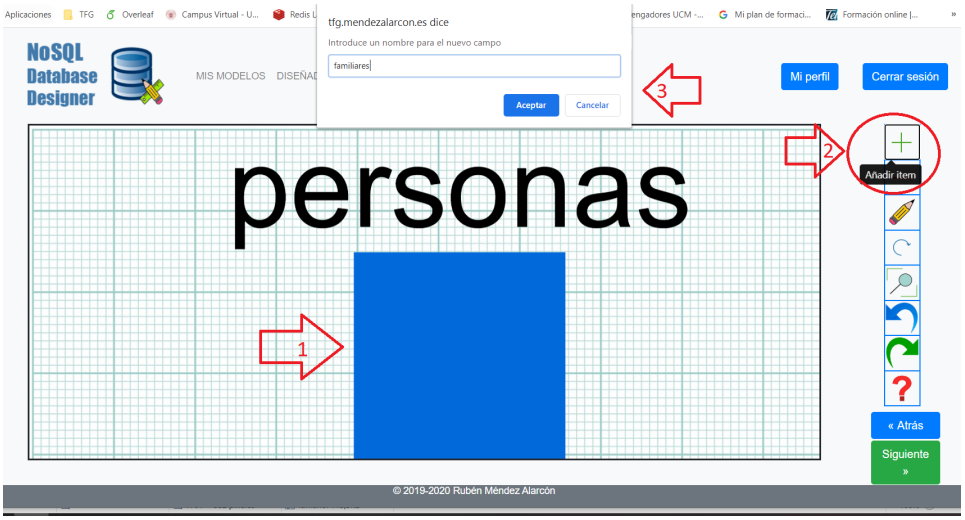

Figura B.15: Pasos a seguir para añadir un item al modelo

que confirmar la operación, pues se borrarán los datos sin posibilidad de recuperación.

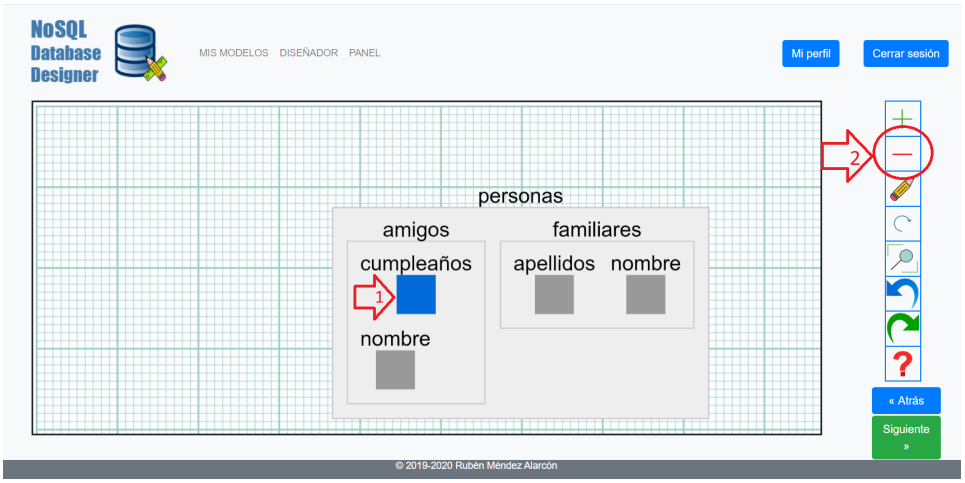

Figura B.16: Pasos a seguir para eliminar un item del modelo

B.5.3. Editar item

La opción de editar item permite cambiar el nombre de un campo o de un agregado. Para cambiar el nombre de un item, es necesario seleccionarlo mediante click y pulsar el botón «editar item». Se abrirá un cuadro de diálogo para introducir el nuevo nombre, que también debe cumplir las restricciones explicadas anteriormente.

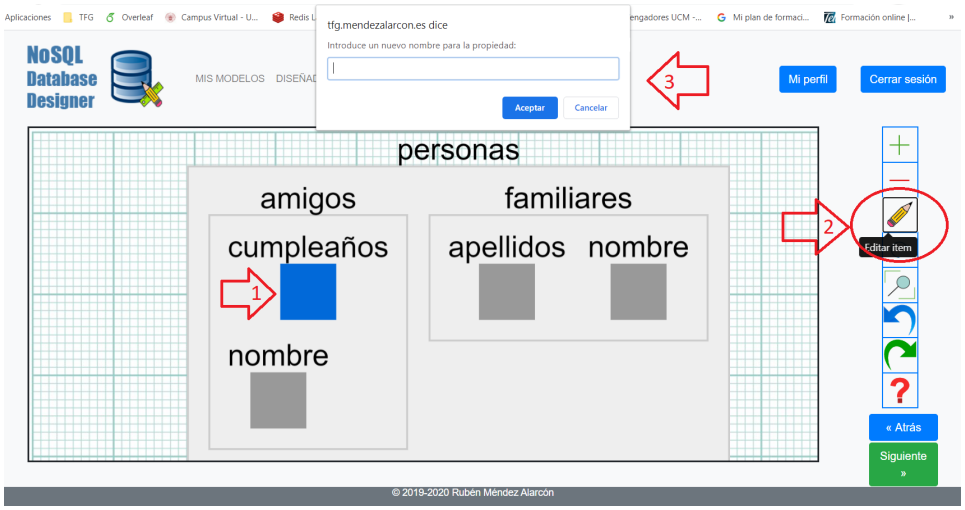

Figura B.17: Pasos a seguir para cambiar el nombre de un item

B.5.4. Reiniciar lienzo

Si se desea descartar el modelo para empezar a diseñarlo de nuevo, no es necesario eliminar el proyecto o eliminar todos los items manualmente. Basta con pulsar el botón «reiniciar lienzo», que devolverá el diseño al estado de un proyecto recién creado. Será necesario confirmar la operación si el modelo ya contiene datos.

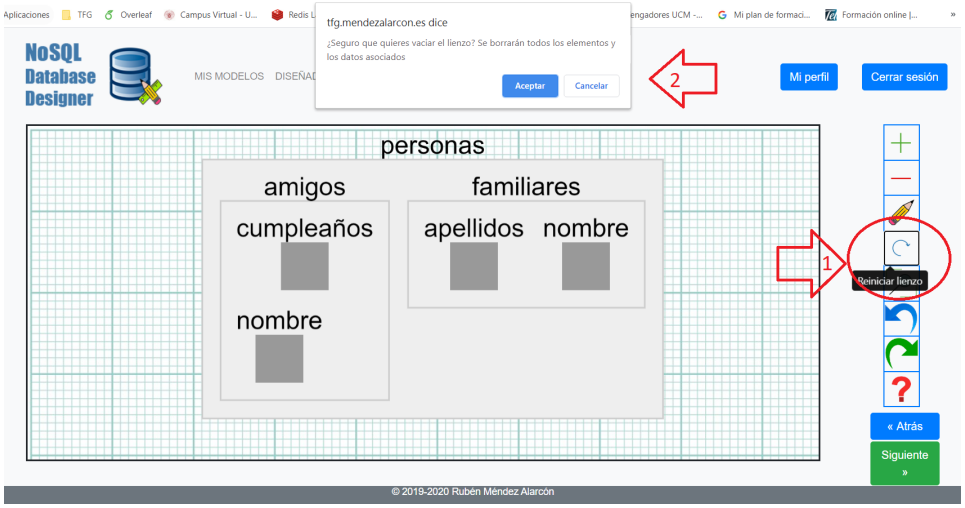

Figura B.18: Pasos a seguir para reiniciar el lienzo

B.5.5. Ajustar

La vista del diseño puede ampliarse y reducirse con zoom, simplemente haciendo scrolling, es decir, con la rueda del ratón. Si se desea volver al estado de zoom original (100 %), la mejor opción es pulsar el botón ajustar.

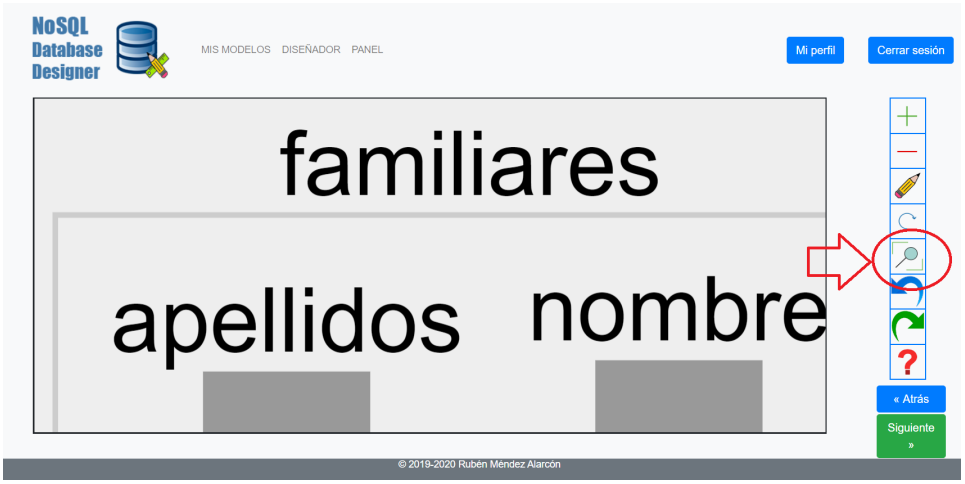

Figura B.19: Ejemplo de uso del botón «ajustar»

B.5.6. Deshacer y rehacer

Los cambios aplicados pueden deshacerse en caso de error o no estar satisfecho con el resultado. Para ello solo hay que pulsar el botón «deshacer». De igual manera, los cambios deshechos pueden rehacerse con el botón «rehacer».

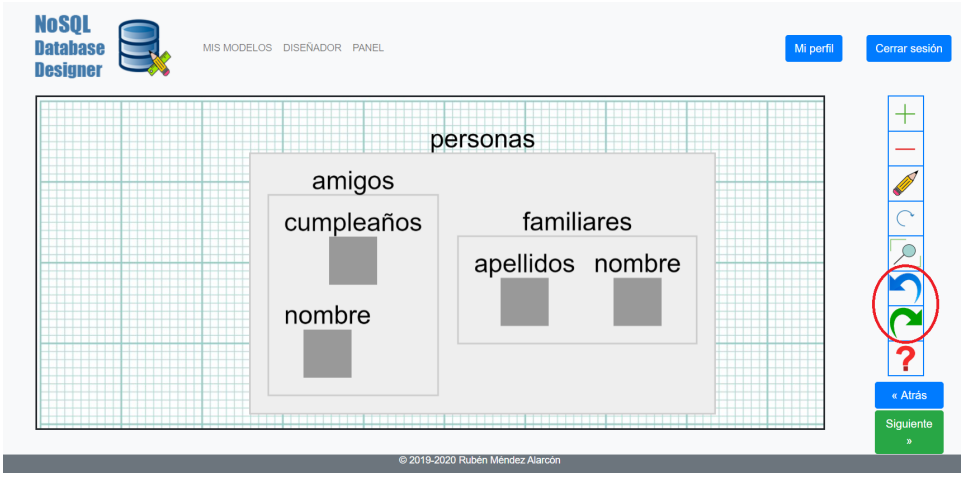

Figura B.20: Ubicación de los botones «deshacer» y «rehacer»

B.5.7. Ayuda

El botón ayuda permite consultar la ayuda del diseñador en cualquier momento. Tan solo es necesario pulsar el botón «ayuda» para lanzar el tutorial de modelado.

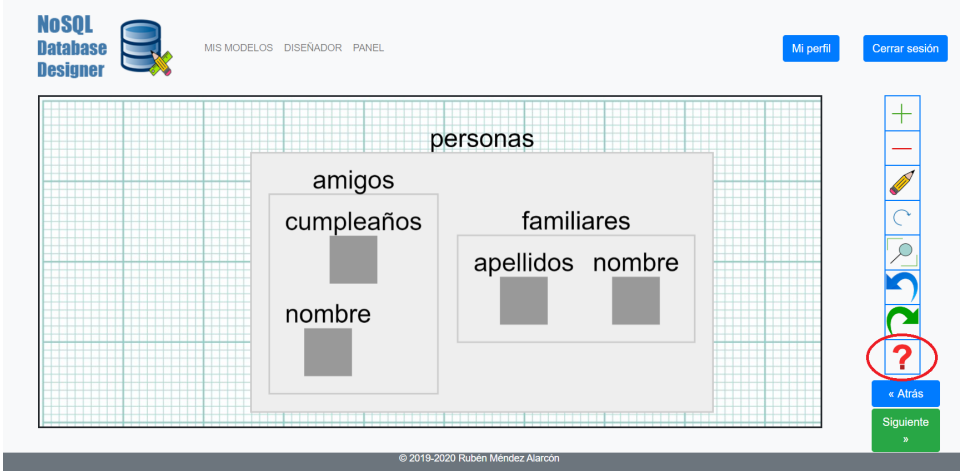

Figura B.21: Ubicación del botón «ayuda»

B.6. Introducción de datos y exportación

Una vez diseñado el modelo, para insertar datos en el mismo se pulsará el botón «Siguiente», situado en la parte inferior derecha de la página del diseñador. En la página de destino aparecerá un formulario con la misma estructura que el modelo diseñado.

B.6.1. Introducir datos

Para insertar datos en el modelo creado simplemente hay que rellenar los campos del formulario y pulsar en «insertar». Si se rellena el formulario con datos erróneos que finalmente se decide no insertar, pueden vaciarse todos los campos del formulario con el botón «restablecer».

Nótese que, al ser el modelo para plataformas NoSQL, no es necesario rellenar todos los campos de cada colección (los documentos finales no tienen por qué tener una estructura homogénea).

B.6.2. Modificar y eliminar datos introducidos

Los documentos insertados pueden consultarse haciendo click en el enlace correspondiente del panel izquierdo de la página. Una vez abierto un

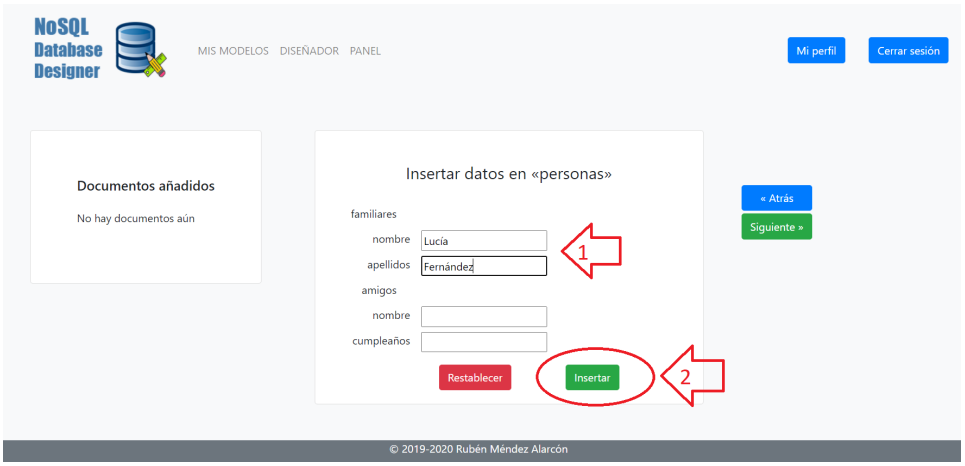

Figura B.22: Pasos a seguir para introducir datos en un modelo

documento, puede modificarse alterando los campos necesarios y pulsando «guardar». Desde esta vista también se puede eliminar el documento pulsando «eliminar».

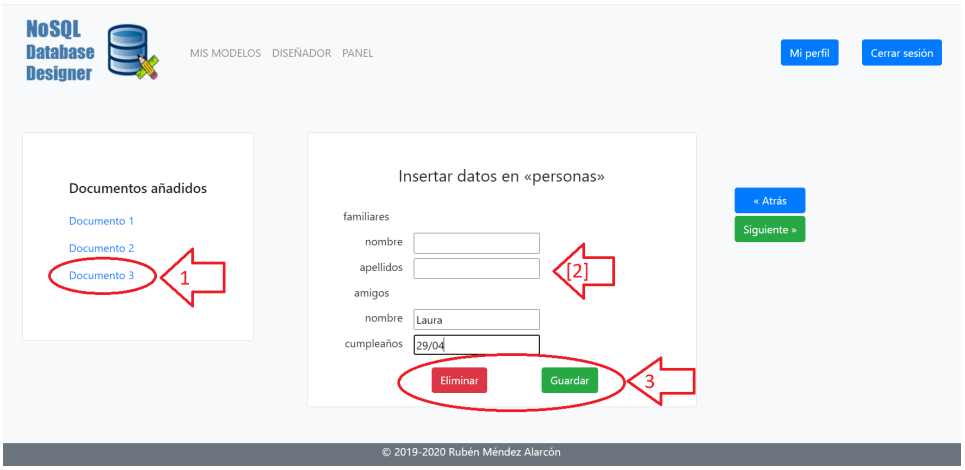

Figura B.23: Pasos a seguir para editar o eliminar datos de un modelo

B.6.3. Exportar resultados

Cuando no se deseen realizar más cambios en el modelo ni añadir información nueva, pueden guardarse y exportarse los resultados. Desde la página de inserción de datos, al hacer click en «Siguiente» se pasará a la página «Guardar y exportar».

Desde esta página se pueden guardar los cambios del modelo (haciendo click

en el botón «guardar»), así como exportar los resultados en forma de código para MongoDB, que se puede copiar fácilmente mediante el botón habilitado a tal efecto.

Figura B.24: Ubicación de los botones «guardar cambios» y «copiar al portapapeles» en la página para exportar resultados

B.7. Cuentas de usuario

En esta última sección se explica cómo utilizar las operaciones asociadas a la información de las cuentas de los usuarios.

B.7.1. Mi perfil

Para consultar la información personal con la que el usuario se registró, simplemente hay que hacer click en el botón «Mi perfil» de la barra superior de navegación. A continuación, se cargará toda la información del usuario. Desde esta misma página se pueden modificar los datos personales, pulsando el botón «Editar perfil».

B.7.2. Editar perfil

Como se comenta anteriormente, haciendo click en «Editar perfil» se pueden actualizar los datos personales del usuario.

Al accionar el botón se cargará un formulario para modificar los datos. También se puede cambiar la contraseña desde dicho formulario, aunque habrá que escribir también su confirmación. Cuando se hayan actualizado los datos, pulsando «Guardar» se harán efectivas las modificaciones

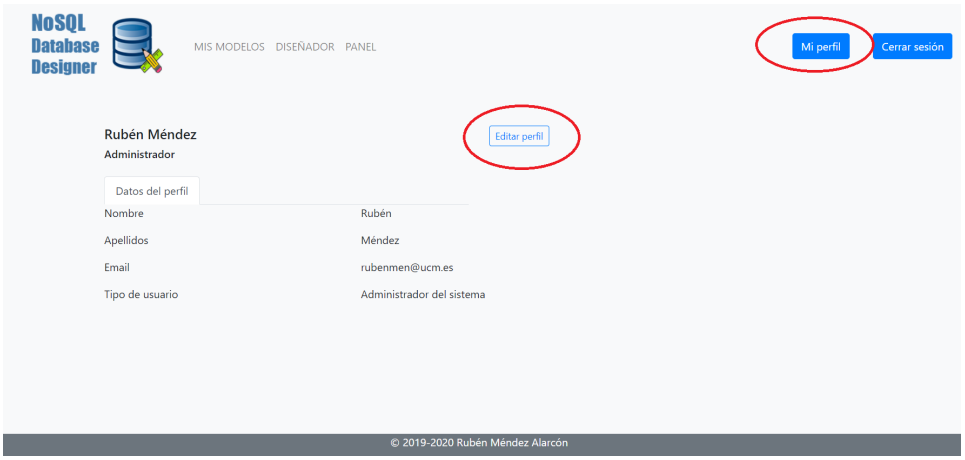

Figura B.25: Ejemplo de perfil de usuario mostrando la ubicación de los botones «Mi perfil» y «Editar perfil»

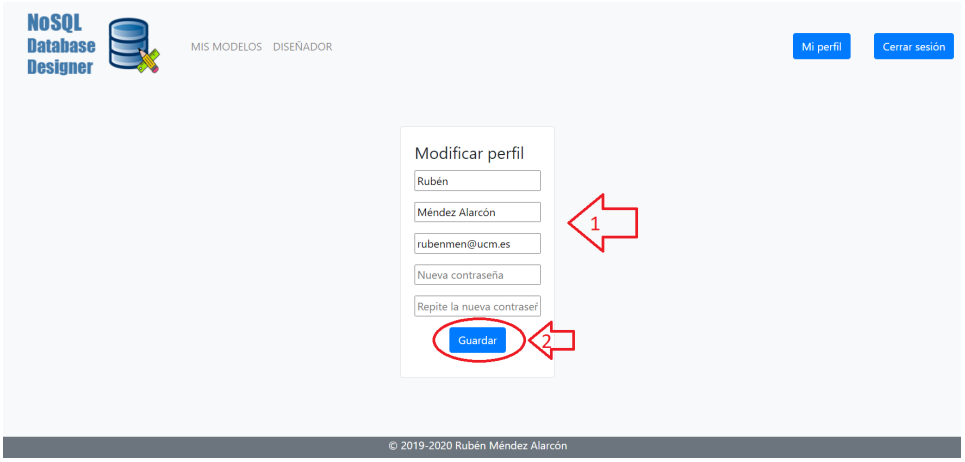

Figura B.26: Pasos a seguir para modificar la información personal del usuario

B.7.3. Administración de usuarios

Para concluir esta guía, en este apartado se introduce la forma de uso de las herramientas de administrador del sistema. Antes de proceder a la explicación, es de interés recalcar que solo un usuario administrador puede otorgar privilegios de administración a otros usuarios. En la base de datos suministrada, el usuario administrador tiene las siguientes credenciales de acceso:

- Email: tfg@mendezalarcon.es
- Contraseña: admin

Una vez que el usuario queda autenticado como administrador, puede dirigirse al panel de administración haciendo click en el botón «Panel» de la barra de navegación. El panel de administración es básicamente un listado donde aparecen todos los usuarios registrados en la aplicación.

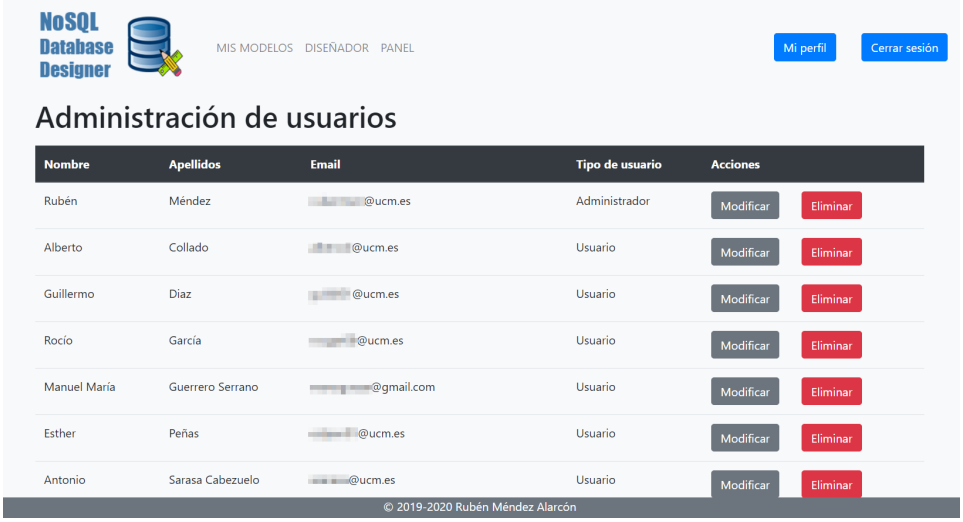

Figura B.27: Vista del panel de administración

Junto a los datos de cada usuario aparece un botón para modificar su información y otro para eliminar a ese usuario. En consecuencia, para modificar un usuario solo hay que hacer click en el botón correspondiente y, como si fuera el perfil propio, modificar los datos sobre el formulario.

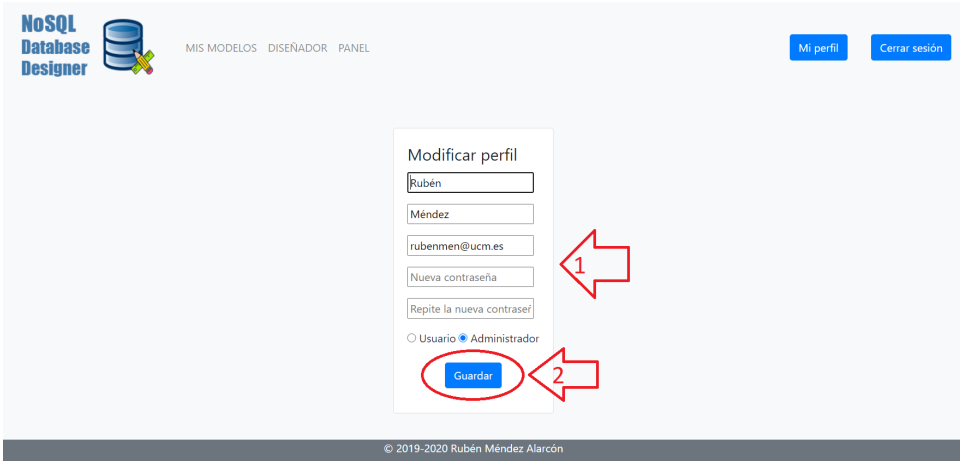

Figura B.28: Pasos a seguir para modificar los datos de un usuario

Para eliminar un usuario, simplemente hay que pulsar el botón y confirmar la operación, como se ilustra en la figura B.29.

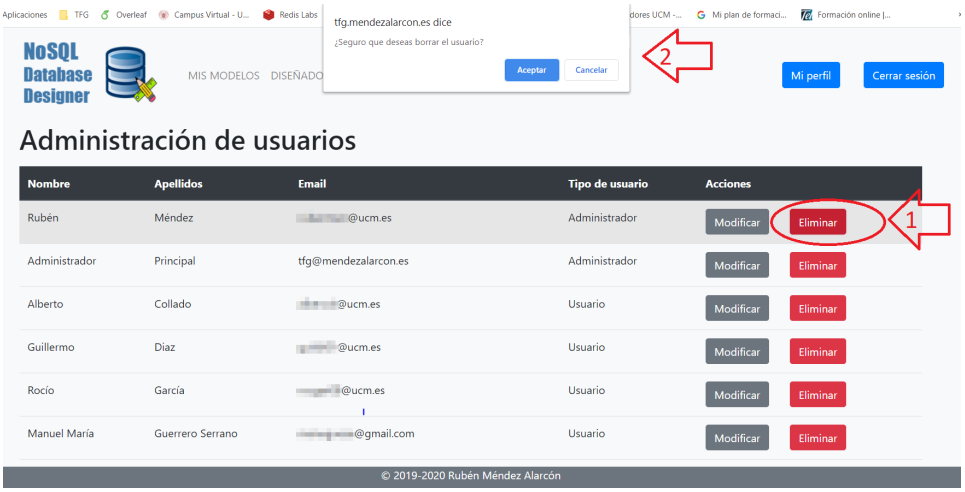

Figura B.29: Pasos a seguir para eliminar un usuario

Apéndice C

Preguntas realizadas en la evaluación con usuarios

C.1. Introducción

En este apéndice se adjuntan imágenes que permiten visualizar, desde el punto de vista de un usuario, las preguntas realizadas en la evaluación con usuarios.

C.2. Preguntas del bloque de información personal

El bloque de información personal se compone de las tres preguntas que se muestran en las figuras C.1 y C.2.

C.3. Preguntas del bloque de conocimientos técnicos

Las preguntas correspondientes a este bloque son las ilustradas en las figuras C.3 y C.4.

C.4. Preguntas del bloque de satisfacción con la aplicación

Las preguntas que recogen la satisfacción de los usuarios con la aplicación son las mostradas en las figuras C.5 - C.9.

C.5. Comentarios

La sección habilitada para comentarios y sugerencias se corresponde con la ilustrada en la figura C.10.

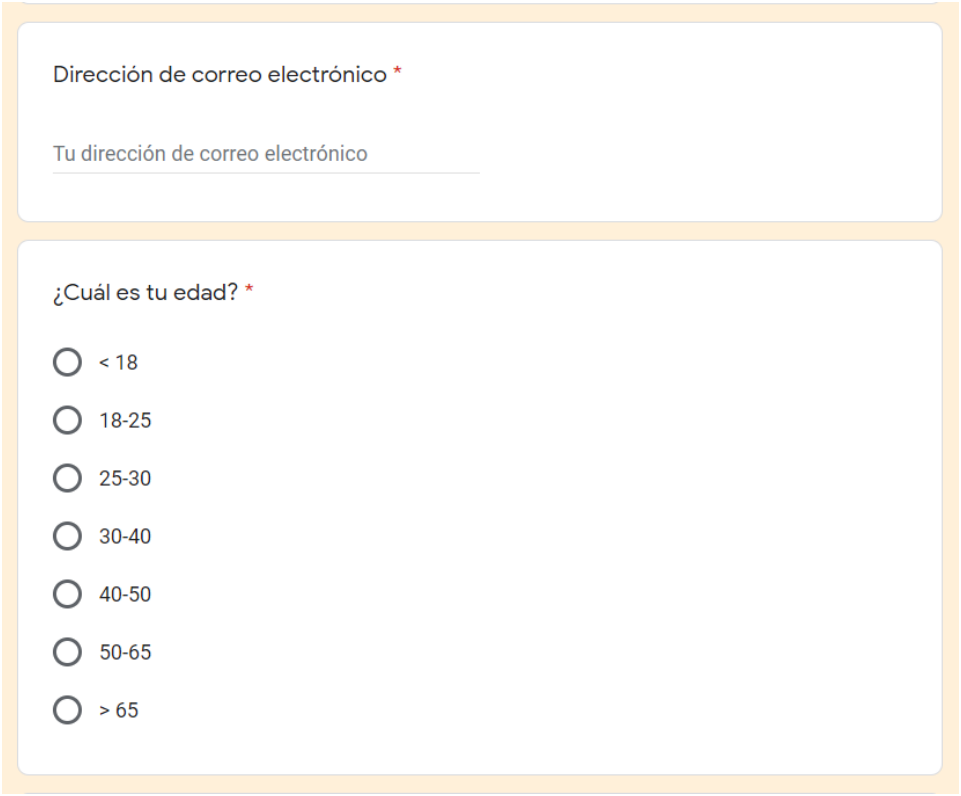

Figura C.1: Pregunta sobre el grupo de edad

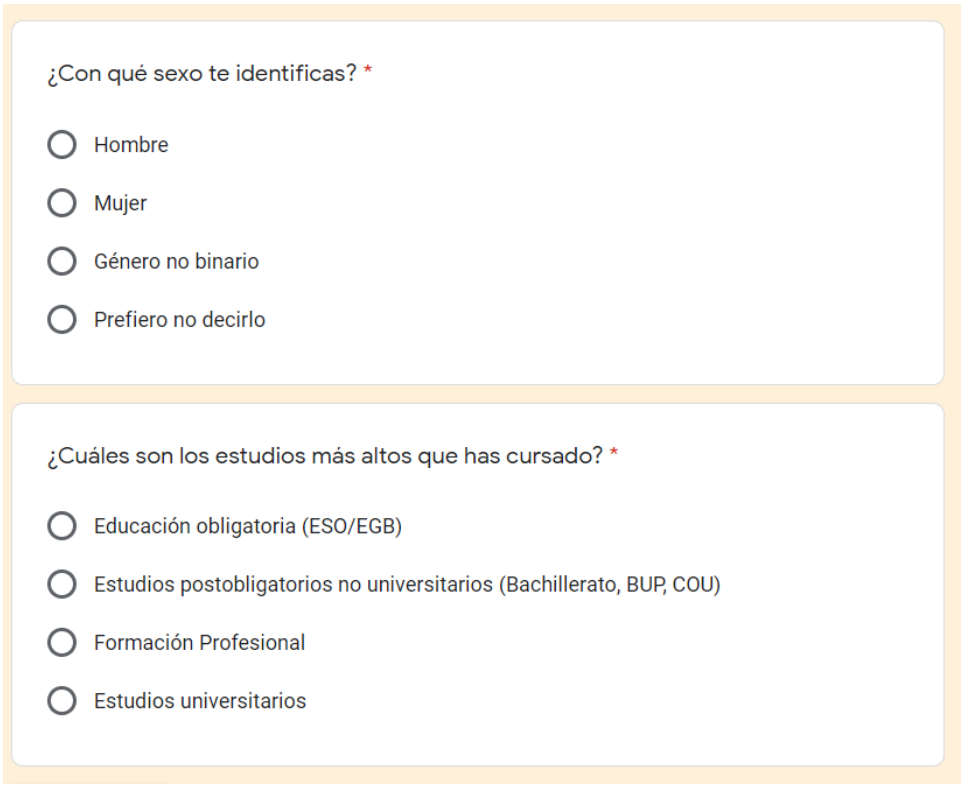

Figura C.2: Preguntas sobre el género y el nivel de estudios

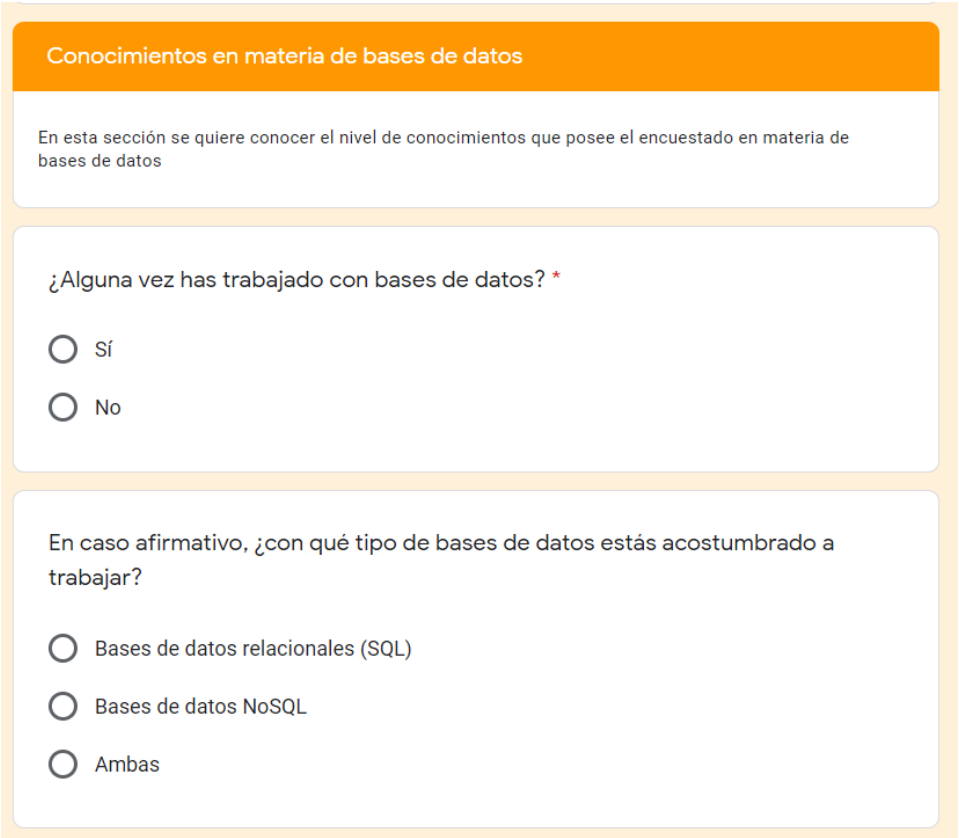

Figura C.3: Preguntas sobre la experiencia con bases de datos

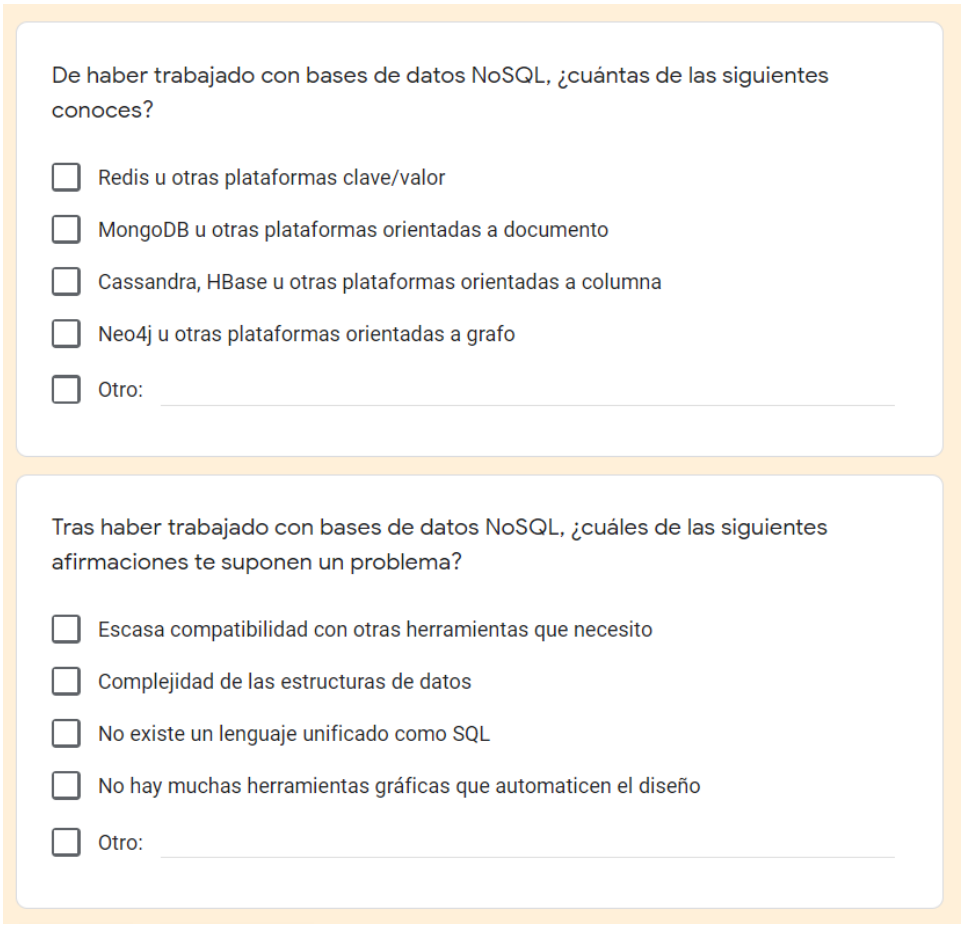

Figura C.4: Preguntas sobre la experiencia con bases de datos NoSQL

Figura C.5: Preguntas sobre la usabilidad I

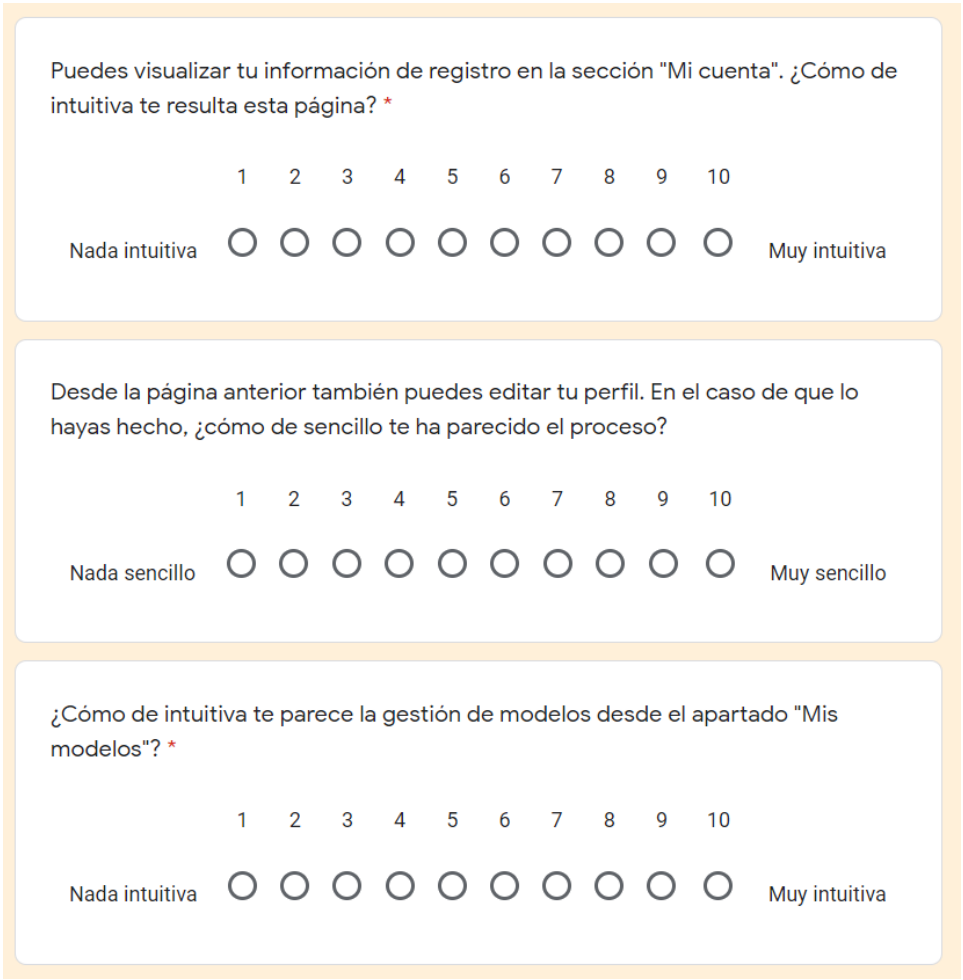

Figura C.6: Preguntas sobre la usabilidad II

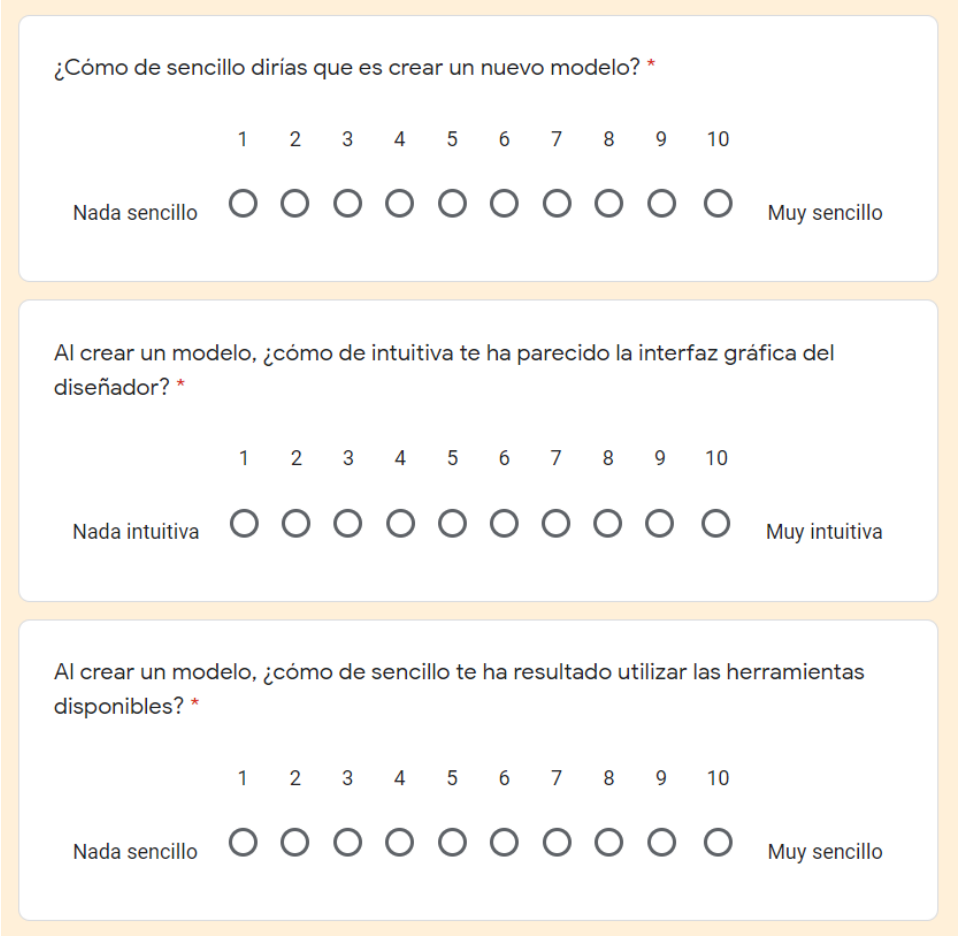

Figura C.7: Preguntas sobre la usabilidad III

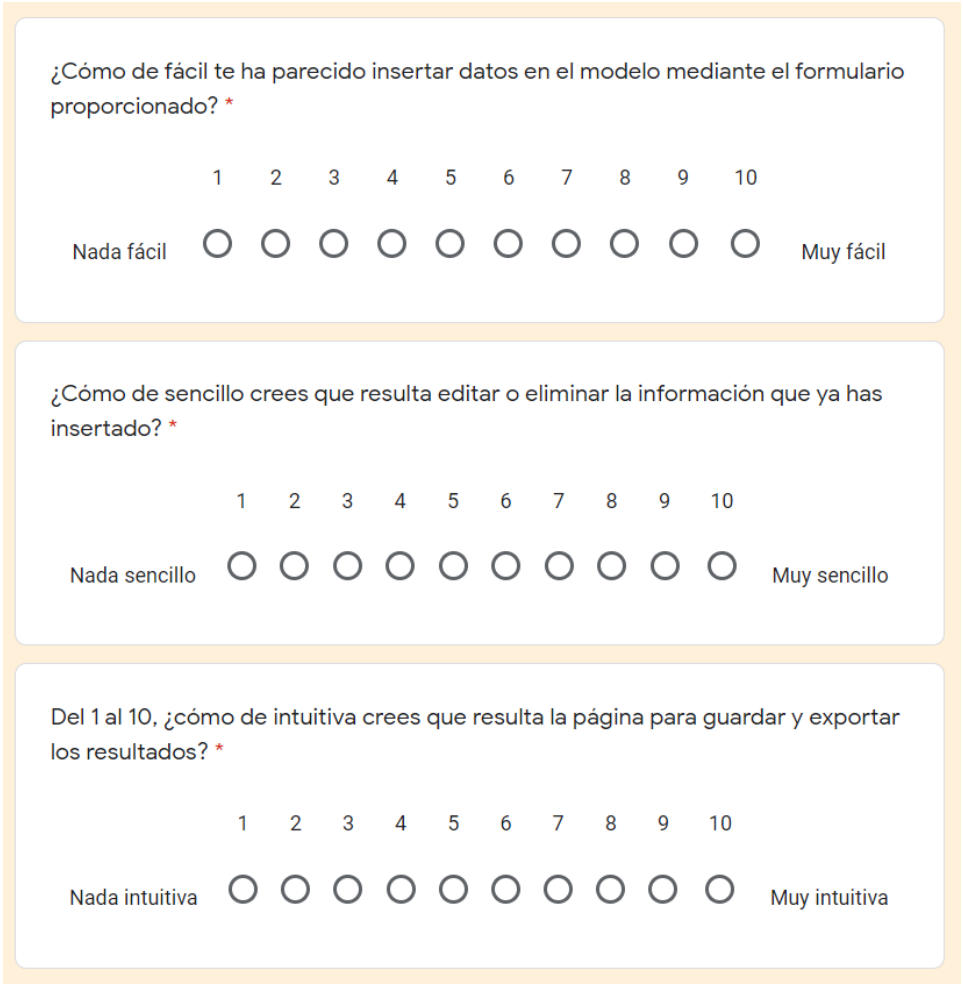

Figura C.8: Preguntas sobre la usabilidad IV

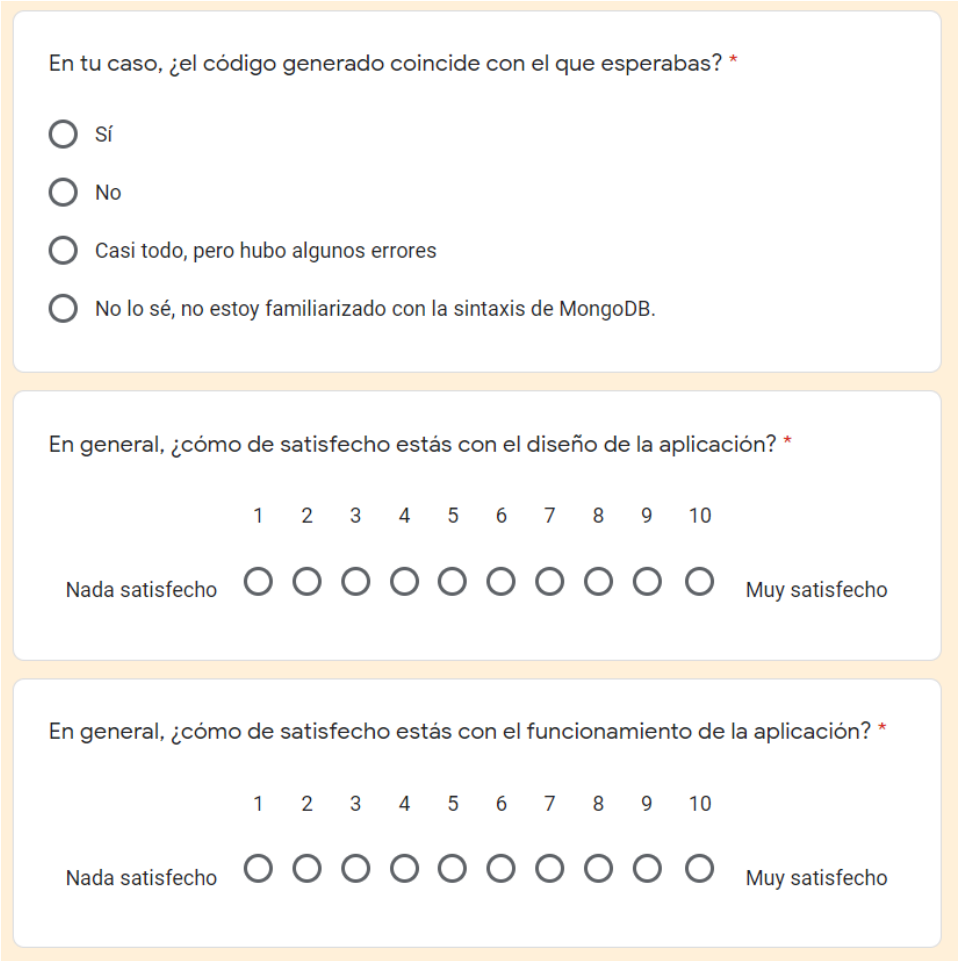

Figura C.9: Preguntas sobre la usabilidad V

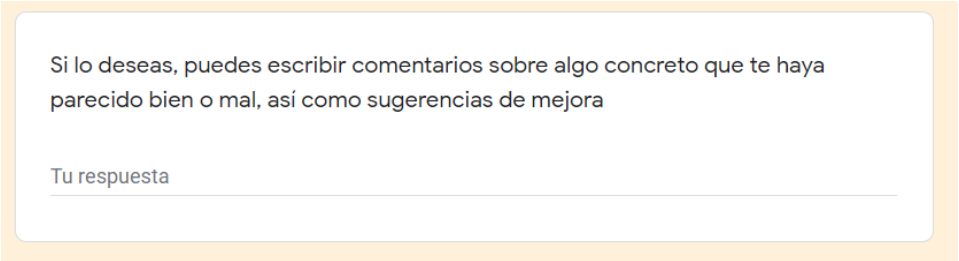

Figura C.10: Sección habilitada para sugerencias y comentarios

Bibliografía

[1] Dayley, B. *Sams Teach Yourself NoSQL with MongoDB in 24 Hours*. Pearson Education, 2015. ISBN 9780672337130.

[2] Sarasa, A. *Introducción a las bases de datos NoSQL usando MongoDB*. Editorial UOC, 2016. ISBN 978-84-9116-266-7.

[3] Cropcho, J. Meet Variety, a Schema Analyzer for MongoDB. 2012. Disponible en `https://www.mongodb.com/blog/post/meet-variety-a-schema-analyzer-for-mongodb` (último acceso, 14 de junio de 2020).

[4] Página de manual de Schema Validation. *MongoDB, Inc.*, 2020. Disponible en `https://docs.mongodb.com/manual/core/schema-validation/` (último acceso, 1 de junio de 2020).

[5] Página de información sobre MongoDB Compass. *MongoDB, Inc.*, 2020. Disponible en `https://docs.mongodb.com/compass/current/` (último acceso, 1 de junio de 2020).

[6] Faulkner, S. et al. HTML 5.2 Recommendation. *World Wide Web Consortium (W3C)*, 2017. Disponible en `https://www.w3.org/TR/html52/` (último acceso, 24 de mayo de 2020).

[7] Información acerca de Bootstrap 4. 2018. Disponible en `https://getbootstrap.com/docs/4.0/about/overview/` (último acceso, 24 de mayo de 2020).

[8] Atkins Jr., T., Etemad, E. J. et al. CSS Snapshot. *World Wide Web Consortium (W3C)*, 2018. Disponible en `https://www.w3.org/TR/CSS/` (último acceso, 25 de mayo de 2020).

[9] JavaScript Guide. *MDN Web Docs*, 2020. Disponible en `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript` (último acceso, 25 de mayo de 2020).

[10] Haverbeke, M. *Eloquent JavaScript: A modern introduction to programming*. No Starch Press, 2014.

[11] Página web de jQuery. 2020. Disponible en `https://jquery.com/` (último acceso, 26 de mayo de 2020).

[12] WWW (jQuery). Entrada: "jQuery". Disponible en `https://es.wikipedia.org/wiki/JQuery` (último acceso, 26 de mayo de 2020).

[13] BALLARD, P. y MONCUR, M. *Sams teach yourself AJAX, JavaScript, and PHP all in one*. Pearson Education, 2008.

[14] FRANZ, M., LOPES, C. T., HUCK, G., DONG, Y., SUMER, O. y BADER, G. D. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, vol. 32(2), páginas 309–311, 2015. ISSN 1367-4803.

[15] DOGRUSOZ, U., GIRAL, E., CETINTAS, A., CIVRIL, A. y DEMIR, E. A Layout Algorithm For Undirected Compound Graphs. *Information Sciences*, vol. 179(7), páginas 980–994, 2009.

[16] WWW (Servidor HTTP Apache). Entrada: "Servidor HTTP Apache". Disponible en `https://es.wikipedia.org/wiki/Servidor_HTTP_Apache` (último acceso, 9 de junio de 2020).

[17] COWBURN, P. ET AL. Manual de PHP. *The PHP Documentation Group*, 2018.

[18] PHP: password_hash. *The PHP Documentation Group*, 2020. Disponible en `https://www.php.net/manual/es/function.password-hash.php` (último acceso, 5 de junio de 2020).

[19] NAYAK, A. *MongoDB Cookbook*. Packt Publishing Ltd., 2014. ISBN 978-1-78216-194-3.

[20] Manual del controlador de MongoDB para PHP. 2020. Disponible en `https://www.php.net/manual/es/set.mongodb.php` (último acceso, 1 de junio de 2020).

[21] Página de manual del método Object.assign(). 2020. Disponible en `https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Object/assign` (último acceso, 6 de junio de 2020).